Biologically Inspired Control Systems for Autonomous Navigation and Escape from Pursuers

Dejanira Araiza Illan
(Candidate)

Dr. Tony J. Dodd
(Supervisor)

Department of Automatic Control & Systems Engineering

July 2012

Summary

This thesis proposes biologically inspired solutions to the problem of autonomous navigation and escape from pursuers for a robot in a dynamic environment. Pursuit can be encountered in real life tasks, but the study of strategies to avoid it has not been extensive compared to the study of pursuit. The problem of navigation and escape has been solved by means of two different available techniques, from a high level control perspective: path planning based on potential functions, and learning navigation strategies with Q-learning. The sensed environment is represented through an approximate cell decomposition or occupancy grid.

Embedding a navigation scheme that allows avoiding static and dynamic possible dangers provides a robot with capabilities that improve its sense of self-preservation when interacting with the environment for the first time, equivalent to innate behaviours in animals. Biologically inspired features have been added to the designs, in the forms of:

- Temporary local goals for the navigation and escape representing possible hideaways (refuges) or locations away from pursuers according to the knowledge of the environment.
- Protean motion towards the goal, to increase the chances of avoiding capture.

Novel contributions include a description of the problem of escape from pursuers, in a dynamic, sensed, previously unknown environment; the use of different kinds of potential functions adapted to the problem, with added biologically inspired proteanism and temporary local goals (hideaways or more distant locations); a comparison of the performance of different potential functions; a comparison of the possible variations in the proposed algorithms; an interpretation of the problem from a game theory perspective; and the use of tabular Q-learning with dynamic parameters and shaped reward functions to find suitable navigation strategies. The work presented in this thesis intends to widen research on the problem of individual escape from pursuers, along with the use of biologically inspired features to solve engineering problems.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Tony J. Dodd, for all the guidance and patience.

All my gratitude to all the people that have encouraged me throughout the PhD:

- My Mum, Dad, my sisters Maria and Gloria, and my boyfriend Chris Kitchen.
- My dearest friends in Sheffield: Javier Caballero, Miriam Flores, Agustín Lugo, Fortino Acosta, Rodrigo Echegoyén, Ernesto Vidal, and the Chacko family (Elishba, Joe and Sophie).
- My university professors in ITESM CCM, particularly Pedro Ponce, Katya Romo, and Rodrigo Regalado.
- All the lovely people from the Mexican and Latin-American Society, PROGRESS, StitchSoc and Son de America.
- Other dear friends in Mexico.

I also would like to thank Charles Winter, for his invaluable collaboration in the implementation of the algorithms in real robots. To the colleagues in the department that shared their knowledge and advice with me.

Finally, I would like to thank my two Mexican sponsors, CONACyT and Becas Complemento SEP, for giving me the great opportunity to study abroad and contribute to the development of science and technology.

Contents

List of Figures

List of Tables

Nomenclature

a	The current action
a'	A future action
a^*	Optimal action that maximises the value function
(a, b)	A cell
$\mathcal{A}$	Set of possible discrete actions
b	Subscript of the allowed actions for a state
b_r	Constant for the force of repulsion away from the predator
B_1, B_2	Constants for the force of attraction towards a prey
BVP	Boundary Value Problem
c_1, c_2	Values for the colour ranges of an image
$\mathbf{C}$	Matrix for the stencils of each cell
$d(.)$	Metric of the Euclidean distance
d_m	Capture distance
d_{min}	Minimum distance from predator to prey
d_s	Distance where the prey shifts to protean motion
d_{sight}	Radius of detection of the prey
d_0	Distance of influence of the repulsion potential function
$danger_p$	Danger given the distance of pursuer p
D	Total danger function
D_1, D_2	Limits of the interval where the predator can chase the prey
D_{grid}	Width of the grid
DFD	Discrete Fréchet distance
e	Subscript of the number of evaders in the observed environment
$e(t)$	Positions of the cells corresponding to the evaders
$\vec{e}$	Unitary vector of random direction for protean motion
$E(t)$	Observed local environment
E^{π}	Expected values given a policy π
$E_1, \ldots, E_m$	Potential functions of each obstacle in the environment
E_g	Potential function of the goal
$f(d(.))$	Function of the Euclidean distance
$f(x, y)$	Function of the Euclidean distance
$f(x_0, y_0)$	Limit of the influence of an obstacle
$f'(.)$	Gradient of the function $f(.)$
$\vec{f}(\vec{r})$	Repulsion force away from the predator
$\vec{F}(\vec{r})$	Attractive force towards a prey
$\vec{F}_{sg}$	Vector from the robot to the subgoal
FIRAS	Force Inducing an Artificial Repulsion from the Surface, in French
$g(d(.))$	Function of the Euclidean distance
$g(y(t))$	Pursuit policy
g_m	Shape or contour of the mth obstacle
$\mathbf{G}$	A matrix to compute the discrete Fréchet distance

$h(a,b)$	Hideaway cells in a grid
$h(x,y)$	A hideaway in a location within a 2-D plane
$h^*(a,b)$	Best hideaways in the grid
i	Subscript of the probable obstacle and pursuer cells, in the same set
i_o	Subscript of the total number of obstacle cells
i_p	Subscript of the total number of pursuer cells
j	Subscript of the navigation steps
k	Number of rotations of the angle θ for the sampling to compute subgoals
k_e	Total number of evader cells in the observed environment
k_o	Total number of obstacle cells in the observed environment
k_p	Total number of pursuer cells in the observed environment
K_{lg}	Effect of the attraction potential of a local goal
K_g	Effect of the potential of a goal
K_i	Effect of the repulsion potential of obstacles and pursuers
K_{LG}	Effect of the reward of the navigation towards the local goal
K_m	Effect of the potential of the mth obstacle
l	Subscript of the dimensions of the space of a potential field
L	Longest link between two curves
LG	Local navigation goal
L_b	Level of brightness of an image
m	Subscript of the number of obstacles in the observed environment
MP	Subset of potential values of adjacent cells
n	Dimensions of the square matrix of the cells of the Taylor series expansion
n_e	Number of evaders in the observed environment
n_o	Number of obstacles in the observed environment
n_p	Number of pursuers in the observed environment
N	Last interval of time of a series
$o(t)$	Positions of the cells corresponding to the obstacles
$\mathcal{O}(\delta^4)$	Terms of order > 4 of the Taylor series expansion
p	Subscript of the number of pursuers or pursuer cells
$p(t)$	Positions of the cells corresponding to the pursuers
$P(a)$	Probability of selecting an action
$P(s', s, a)$	Transition probability to state s' given s, a
P_0	State transition probability kernel
$P_{(a,b)}$	The set of probabilities of a cell
$P_{(a,b)}(F_s)$	Probability of a cell of being free space
$P_{(a,b)}(o)$	Probability of a cell of being an obstacle
$P_{(a,b)}(p)$	Probability of a cell of being a pursuer
q_l	Largest number of points in a path, q_1 or q_2
q_1	Total number of points in path 1
q_2	Total number of points in path 2
$Q^{\pi}(s,a)$	Action-value function for a policy π
$Q^*(s,a), V^*(s)$	Optimal policies

r	Power for the potential function of a solid obstacle
$r(t), r$	Reward due to the transition of state in time step t
r_{det}	Radius of detection space of a prey
r_{sg}	Radius for the computation of subgoals
$\lvert\vec{r}\rvert$	Distance from predator to prey
$R(t)$	Discounted reward over time
$\mathbb{R}^n$	Space of n dimensions
R_c	Punishment of the navigation close to obstacles and pursuers
R_g	Square of the distance between a point in a 2-D plane and the goal
R_{LG}	Reward of the navigation towards the local goal
R_t	Total reward function
$RMSE$	Root-mean-square error
s	Current state
s'	Next state
s_0	Starting state
(sg_x, sg_y)	Location of the subgoals in the 2-D plane
$\mathcal{S}$	Set of possible discrete states
S_m	An obstacle represented by a solid shape
t	Discrete instant of time
u	Subscript of the neighbour configurations to current (j)
$\hat{u}$	Unitary direction vector towards the subgoal
$U_a(a, b)$	Quadratic attraction function exerted by the local goal
$U_r(a, b)$	Modified FIRAS repulsion function exerted by obstacles and pursuers
$U_t(a, b)$	Sum of the attraction and repulsion potential functions
v	Point in the trajectory of the predator
v_j	Current location in the navigation
v_{j+1}	Next location in the navigation
v_p	Velocity of the pursuers, relative to the robot
v_r	Velocity of the robot
V	Set of neighbour configurations for the steepest descent search
$V^{\pi}(s)$	Value function for the states given a policy π
$\vec{V}$	Perturbation
$w_h(a, b)$	Function for each cell of the intersection of shadows
$w_d(a, b)$	Function for each cell of the combined distances of all pursuers
$w_d^*(a, b)$	Farthest location away from pursuers in the local environment
x_e	Location of evader e
x_{max}	Maximum displacement of the robot
x_p	Location of pursuer p
(x, y)	Point in a 2-D space
$y(t)$	Collected information at a discrete time instant
$z_{a,b}$	Value of each cell (stencil) from the Taylor series expansion in a grid
$\vec{z}$	Set of equations for all the values in the grid
$\vec{z}_{new}$	Result from the iterative numerical method

$\vec{z}_{old}$	Previous values in the iterations of the numerical method
α	Learning rate
α_{GD}	Step size for the gradient descent method
γ	Discount rate for the reward
δ	Displacement or distance of the Taylor series expansion
ϵ	Limit for the probability of choosing an action (ϵ-greedy)
ϵ_p	Strength of a perturbation
ϵ_{th}	Threshold for the convergence error in the numerical methods
θ	Angle for sampling to compute subgoals
θ_r	Angle of the robot from the horizontal
λ	Weight for the contribution of the rewards at every time step
λ_{SOR}	Parameter for the SOR iterative method
μ	Media of a series of numbers
π	Policy or series of actions according to the states
ρ	Immediate reward function of a pair action-state
τ	Temperature for exploration in soft-max action selection mechanism
$\phi(a,b)$	Value of the potential in a cell of the grid
$\phi(x,y)$	Continuous potential field
ϕ_{goal}	Boundary value for the local goal
ϕ_o	Boundary value for the obstacles
ϕ_p	Boundary value for the pursuers
$\phi_{wf}(a,b)$	Wave-front potential function when expanding from the pursuers
ψ	Basis functions in function approximation
ω	Weights of the basis functions in function approximation
ω_p	Angle between the p pursuer and the robot
∇	Gradient operator
∂	Partial derivative operator

Chapter 1

Introduction

This thesis addresses the problem of autonomous navigation and escape from pursuers in mobile robotics, present in real life applications and more complex than collision avoidance only. The studied perspective of the problem contemplates the lack of previous knowledge about the environment, or the unavailability of models that describe the environment, also considering the sensing as the only source of information. The problem of escape from pursuers (also called evasion in a pursuit-evasion game context) has not been extensively studied as its counterpart, the pursuit.

The proposed designs in this thesis that compute solutions to the problem represent two different points of view: the process of path planning with potential functions, and the computation of strategies from reinforcement learning, considering their advantages to solve the particular problem of navigation and escape with limited information. Biologically inspired features have been incorporated into the designs, following a current trend in robotics and other engineering areas of turning eyes to nature and its processes, to find better solutions as an alternative to traditional engineering solutions. The features offer a complement and alternatives to avaliable autonomous navigation theory and algorithms, in the escape of pursuers. Different variations are considered for the designs, being evaluated through the analysis of simulations, and the feasibility of implementation in a robotic platform.

The novel contributions include: a definition of the problem of navigation and escape from pursuit with limited information; the review of different areas of knowledge focusing on their fusion in the design of high level solutions to the problem, including bio-robotics, biology, path planning, pursuit-evasion games, and reinforcement learning; the integration of biologically inspired features in popular path planning and reinforcement learning methods; and an evaluation of the proposals by a statistical analysis of their success.

This chapter presents the motivation behind the project contained in the thesis (Section 1.1), an introduction to the problem of navigation and escape from pursuers (Section 1.2), the aims and objectives (Section 1.3), the main novel contributions (Section 1.4), a list of publications resulting from the work in the thesis (Section 1.5), and an outline of the contents in the rest of the chapters (Section 1.6).

1.1 Motivation

Achieving the autonomous navigation of a robot is an ongoing problem, along with the demands of the particular task, the characteristics of the elements in the environment, the physical and kinematic limitations of the robotic platform, and the architecture of the control system (Choset et al., 2005). Robots with autonomous navigation capabilities can be applied to numerous areas of service like exploration, surveillance, recognition, rescue, collection, and transportation. In some environments, threats can be present in the form of pursuers or pursuit systems. Ideally, autonomous robots should be able to deal with the different circumstances that they encounter during their navigation task, trying to self-preserve their integrity within the limits of their capabilities. Added to the problem of pursuit is the lack of information in the beginning of a navigation task, where no previous mapping, learning or modelling of the environment has taken place, but the robot still needs to respond to the threats in the environment.

This problem of a first interaction with pursuers and local available information coming from sensing has not been exhaustively explored, as collision avoidance (Choset et al., 2005), or even the problem of pursuit (Isaacs, 1965). Nevertheless, reliable autonomous systems, capable of making the best of their circumstances, are needed in some tasks. In order to avoid the danger posed by pursuers, a set of actions should provide an adequate response according to the available knowledge. Many proposed control systems for autonomous navigation do not contemplate any other action when interacting with the general environment than collision avoidance, for both static and dynamic obstacles.

A gap in the design for control systems to compute navigation strategies or paths that contemplate a response to other kind of threats, like pursuers, is present within the general field of autonomous navigation, particularly for partially observable or incomplete information environments. Some of the few efforts attempting to solve the problem of navigation and escape from pursuers, found during the literature review, include: Amin et al. (1997), Karaman and Frazzoli (2010), and Masoud (2002, 2003a,b) from a path planning perspective, mostly based on potential functions; and Leondes and Mons (1979), Yavin (1986), de Villiers et al. (1987), and Fuchs et al. (2010) in the context of pursuit-evasion games. The proposed approaches consider a full knowledge of the actions of the pursuers, their capabilities, and the environment, in general. However, the design of escape (or evasion) strategies for autonomous robots can be performed independently from the design of pursuers, and without necessarily committing the design to a particular pursuit strategy.

The incorporation of biologically inspired features in robotics is an area of opportunity in research (Bekey, 1996; Habib et al., 2007). Biologically inspired features can be used for the improvement of solutions to the problem of autonomous navigation, towards its robustness when the environment becomes challenging. Studies on anti-predator behaviour modelling like Broom and Ruxton (2005) or Furuichi (2002), or even computational extensions of general phenomena like the results of co-evolution in Nolfi and Floreano (1999), offer some inspiration for proposals to achieve autonomous navigation and escape from pursuers.

The capability of reacting to threats in the form of pursuers, with information from sensing only, is investigated in this thesis. Mechanisms to compute paths or navigation strategies that implement biologically inspired self-preservation behaviours, like seeking refuge or protean motion, as part of the navigation against collisions, are proposed. The areas of artificial intelligence, control systems and robotics offer numerous options to solve autonomous navigation problems, for example basic reactive controllers (Kim and Shim, 2003), high level plans (LaValle, 2006; Choset et al., 2005), or learning strategies (Sutton and Barto, 1998). This thesis addresses the computation of high level paths or navigation strategies by extending popular approaches from path planning with potential functions, and later through reinforcement learning, to extend previous work on a more general autonomous navigation (based on collision avoidance) to contemplate the escape from pursuit. Situations of navigation whilst being pursued involve several elements that are out of control of the designer, like the capabilities of the pursuers, and the characteristics of the environment. Thus, designing general control systems that are able to deal with limited information, and for a wider range of threats, are highly desirable to increase the autonomy of robots.

1.2 Problem definition

Autonomous robots that navigate in unknown, dynamic environments need to make decisions according to the information they perceive from their sensing devices. When a robot is deployed to a new place that contains dynamic threats in the form of pursuers, along with obstacles, an adequate response is needed. A set of navigation actions that avoids collisions, and also capture by pursuers, provides a first response in the self-preservation of a robot, regardless of any other learning or control system as a consequence of the interaction with the environment, or according to other objectives of the navigation task. The design of such a control system that provides actions to navigate and escape from pursuers needs to consider the restrictions in the knowledge of the environment, as well as the unavailability of dynamic models of the pursuers and obstacles, and the stochasticity of real environments.

The problem of autonomous navigation and escape from pursuers is encompassed within the family of general autonomous navigation, and consists of the following elements: *(a)* a locally observed environment (partially observable or with incomplete information); *(b)* obstacles; *(c)* pursuers; and *(d)* the robot that performs the navigation. A plausible solution is the computation of paths or navigation strategies based on adequate techniques according to the available information, the chosen representation of the environment, and the chosen control approach. For this matter, path planning methods like potential functions combined with a cell decomposition representation, or tabular Q-learning with discrete actions and states are considered alternatives, based on their comparative advantages over other path planning methods and solutions to pursuit-evasion games.

1.3 Aims and objectives

General aim

The general aim of this thesis is to design solutions to the problem of navigation and escape from pursuers, according to a given definition, using combinations of path planning and reinforcement learning with biologically inspired features. A solution to the problem is:

- A navigation path or strategy that avoids collisions with obstacles; and
- A navigation path or strategy that leads away from pursuers, and prevents capture for as long as possible.

Specific objectives

- Conducting a comprehensive literature review in the following topics:
  - Biologically inspired robotics, particularly the current trends and projects, the degrees of biological inspiration, and examples of applications of inspiration.
  - Behavioural and mathematical models of anti-predator motion mechanisms for general taxonomies of animals, and their extension to computational applications.
  - Path planning, particularly the method of potential functions, dealing with dynamic environments, and applications to the task of escape from pursuers or other similar problems.
  - Pursuit-evasion games, considering the proposed methods to solve them, with emphasis on evasion strategies.
  - Reinforcement learning for autonomous navigation.
- Integrating concepts of path planning, reinforcement learning, game theory and biology to propose solutions for autonomous navigation and escape from pursuers.
- Proposing a particular methodology for the analysis of the environment for the specifications of the problem, and then the computation of a plan or navigation strategy according to the analysis, from path planning with potential functions, and reinforcement learning.
- Proposing solutions to the problem of navigation and escape that are usable for different robotic platforms, and for dynamic, stochastic environments.
- Evaluating the performance of the proposals through simulations and a statistical analysis.

- Designing a visualisation to appreciate the results of the different steps in the proposed algorithms, in MATLAB.
- Demonstrating some of the proposals in a real robotic platform.

1.4 Main original contributions

- Definition of the problem of navigation and escape from pursuers, for a sensed environment where no previous model is available. The particular definition presented in this thesis addresses the problem of navigation and escape from pursuers in a novel form: the case where no previous information about the environment, and no model of the pursuer are available. This perspective of the problem has not been studied as extensively as more general autonomous navigation for collision avoidance, nor the counterpart of evasion: the pursuit. The definition of the problem from the perspective of path planning is mentioned in Chapter 4, and from the perspective of game theory solved by reinforcement learning in Chapter 7.
- A review of different areas of knowledge: behavioural and mathematical biology, path planning, pursuit-evasion games, and reinforcement learning, oriented to finding models and previously proposed solutions to the evasion or escape from pursuers in dynamic, unknown environments and applied to basic robotic platforms. The review highlights the novelty of the study of anti-predator navigation motion and its application to robotics, as well as a wide range of possibilities to solve the problem of navigation and escape from pursuers using different methods of path planning, machine learning, probabilistic analysis, optimisation theory, intelligent control, and many others. The review is presented throughout Chapters 2, 3, and 6.
- The use of inspiration from biological, individual, general anti-predator behaviours (but not necessarily mimicry) in path planning and reinforcement learning to solve the problem of navigation and escape from pursuers. The integration of biologically inspired features with both path planning with potential functions and basic reinforcement learning algorithms, takes the form of the design of a high-level control system that computes a path or a strategy to achieve the evasion and navigation avoiding collisions. Specifically in the designs, the following contributions can be outlined:
  - A biologically inspired analysis of the environment to compute the best location for the robot in the current situation, from the sensed environment, and using the concept of refuges (Chapter 4).
  - An implementation of biologically inspired proteanism through the use of navigation subgoals (Chapter 4)
  - The incorporation of pursuers and the proteanism (subgoals) into popular potential functions (Chapter 4).

  - Shaped reward functions from the biologically inspired analysis of the environment and parameters for the reinforcement learning algorithm (Chapter 7).

  The contributions in the designs are the combination of concepts from different areas of knowledge and applications, in a novel form to solve the problem of navigation and escape from pursuers.

- An evaluation and comparison of the performance of different alternatives for the computation of navigation plans or strategies for the navigation, from path planning and reinforcement learning techniques. The proposed variations in the algorithms that compute the paths or strategies regarding the use of different potential functions and the parameters in the designs, are evaluated and compared using a statistical analysis. The performance of the proposals has been evaluated through larger sets of simulations with variations of the fundamental parameters, leading to strong conclusions about the results. This proposed methodology based on statistics contributes to set the basis for future evaluations of generated paths in different applications of autonomous navigation (Chapters 5 and 7).

1.5 Publications

The work in this thesis has contributed to the following publications:

D. Araiza-Illan and T.J. Dodd. Biologically inspired controller for the autonomous navigation of a mobile robot in an evasion task. In *World Academy of Science, Engineering and Technology*, 68: 780-785, 2010.

D. Araiza-Illan and T.J. Dodd. Bio-inspired autonomous navigation and escape from pursuers with potential functions. In *Proceedings of Towards Autonomous Robotic Systems*. In press, 2012.

1.6 Thesis outline

The thesis is structured to provide the literature review and theoretical basis in Chapters 2, 3 and 6, and the main research related to the theory in Chapters 4, 5, 7 and 8. Finally, the concluding remarks are presented. A more detailed description is presented next.

Chapter 2 starts by presenting a general review of the fields of bio-robotics, mentioning different forms and degrees of biological inspiration applied to solve engineering problems. Several examples of current projects are used to illustrate the wide variety of implementations of biological inspiration. The chapter also presents a review of literature on prey-predator interactions from a mathematical and behavioural perspective, including general individual anti-predator behaviours encountered in different species, different mathematical models of individual anti-predator actions, other models of prey-predator interactions from different perspectives, computational implementations of the models and their application to other different problems. Particular interest has been paid to individual models of anti-predator behaviours that can be used in robotic applications like autonomous navigation.

Chapter 3 continues the literature review, focussing on the field of path planning for navigation of autonomous robots. The main methods used for path planning are surveyed in general terms, with particular interest in solutions to the problem of escape from pursuers, and closely related areas. The chapter focuses on potential functions, and different proposed variations in their implementation, due to their particular advantages.

Chapter 4 presents the path planning problem of navigation and escape from pursuers that is solved in this thesis, altogether with an overview of the decisions made with respect to the designs of solutions (including the control approach, the techniques, and the architecture). Subsequently, the designs for the analysis of the environment and path planning are presented. A biologically inspired analysis of the environment is proposed, based on the concepts of refuges and proteanism (randomness in the navigation) from the literature presented in Chapter 2, and according to the limits of the defined problem. The analysis computes an occupancy grid of the elements in the environment, and a local navigation goal, which are mapped into a potential function. Different kinds of potential functions selected from the literature review in Chapter 3 are tailored for the solution of the problem of navigation and escape, and finally a search method is proposed to compute a path that would lead to avoiding collisions and escaping from pursuers.

Chapter 5 evaluates and compares the proposed variations for the path planning designs, through simulations that demonstrate the performance of the proposed analysis of the environment and path planning in a visual manner, but also through a statistical analysis of the results. Different kinds of simulations were performed for the variations in the analysis of the environment (regarding mainly the implementation of the proteanism and the characteristics of the grid), the different kinds of potential functions, and possible elements in the environment that complicate the problem. The analysis of the results provides the best alternatives, from the ones proposed and evaluated, to avoid the capture (or delay the capture) according to the characteristics of the environment.

Chapter 6 returns to the literature review, presenting an overview of the main concepts of pursuit-evasion games, and the main groups of proposed solutions to such games. The computation of evasion strategies independently from pursuit strategies, and their application to autonomous navigation in robotics are of particular interest. Subsequently, the chapter presents a summary of the main concepts of reinforcement learning, including the characteristics of the algorithms and relevant implementation considerations. Some applications of reinforcement learning to autonomous navigation are briefly mentioned, as reinforcement learning is of interest for the final designs of solutions to the problem of navigation and escape from pursuers, when mathematical models of the environment and the other elements (like the pursuers) are not available.

Chapter 7 presents an interpretation of the same problem defined in Chapter 4 from a game theory perspective, and a solution by means of a reinforcement learning algorithm. A biologically inspired implementation of tabular Q-learning is presented, based on the same analysis of the environment proposed in Chapter 4, and adding a shaped reward function (also with bio-inspired concepts) and dynamic learning parameters. The selection of the learning parameters is justified through an analysis of their effect on the computation of an adequate solution to the problem of navigation and escape. In the last part of the chapter, an evaluation of the proposed design is presented, through a statistical analysis of simulations similar to the ones presented in Chapter 5.

Chapter 8 describes the implementation process of some of the designs in a robotic platform. A qualitative evaluation is presented, highlighting relevant aspects of the sensing and the low level control used to complement the designs presented in Chapter 4.

Chapter 9 summarises the content of the thesis, with attention to the accomplishment of the main objectives stated in the introduction. The main contributions and results for the chapters that present and evaluate the designs are outlined, followed by the global achievements. Finally, several ideas on possible future work are introduced.

Chapter 2

Bio-robotics and biological studies on anti-predator behaviours

Biological features can serve as an inspiration to solve robotic problems, offering solutions to the problem of autonomous navigation and escape from pursuers. This chapter presents a literature review on the fields of bio-robotics, biological anti-predator behaviours, prey-predator mathematical models and some applications, to define the biologically inspired basis of the core work in this thesis. Bio-robotics is an emergent area of study that has broadened and acquired importance in the last twenty years, due to the applicability and flexibility of bio-inspiration usage. Anti-predator and prey-predator interactions are biological features that have been studied in many different fields including behavioural biology, mathematical biology, evolutionary biology, game theory, and control systems.

An explanation of the main subareas of study in the field of bio-robotics is offered, to clarify the existence of different degrees of bio-inspiration. Section 2.1 presents a definition and historical review of bio-robotics and its degrees of biological inspiration, biomimetics and biologically inspired robotics. Many examples of recent bio-robotics projects are presented, for both biomimetics and biologically inspired degrees, to facilitate the illustration of different degrees of inspiration.

An overview of relevant and important biology related studies on anti-predator behaviours and prey-predator interactions is presented subsequently in Section 2.2. More specifically, Subsection 2.2.1 deals with general anti-predator behaviours. Subsection 2.2.2 summarises population dynamics models, game theory fusions and food web models. Subsection 2.2.3 presents other anti-predator and prey-predator interaction general models. Subsection 2.2.4 summarises evolutionary extensions of the models in population dynamics and game theory fusions. Subsection 2.2.5 summarises some interesting work on enhanced and applied anti-predator behaviours.

Finally, Section 2.3 analyses the reviewed literature from a critical point of view, highlighting the relevance and orientation of anti-predator behaviours that has been considered for the designs that solve the problem of autonomous navigation and escape from pursuers.

Section 2.4 presents the conclusions of the chapter.

2.1 Bio-robotics

Bio-robotics is a relatively recent area of research, mainly motivated by the idea that combining biology or bio-inspired concepts and engineering is advantageous, as biological systems have grown to be more efficient through evolution, particularly in terms of energy. Bio-inspired robotic systems are closely interrelated with other areas of knowledge and research, such as behavioural biology, cybernetics, nanotechnology, control systems, and mechanical engineering. The developments that have been taking place in the area of bio-robotics have resulted in robotic platforms to study and understand biology, as well as robotic solutions to engineering problems that incorporate biological tools and mechanisms. Regardless the degree of biological inspiration that is used in a design, which sets the boundaries within the classification of bio-robotics into biomimetics or biologically inspired robotics, any project involving a fusion between biology and robotics is considered within the field of bio-robotics.

A general definition of bio-robotics is the incorporation of biological ideas and mechanisms into robotics. Bio-robotics is divided into two main classes according to the degree of biological inspiration and the main objective of the fusion of biology and robotics in the system: biologically inspired robotics, and biomimetics. The definition of each class is presented in Subsections 2.1.1 and 2.1.2, along with several illustrative examples.

Different degrees of bio-inspiration can be observed in the literature, ranging from exact mimi-cry of animal processes to abstract metaphors. The term *bio-inspired* has been defined by some authors in more general terms (Yuan and Yang, 2009), contemplating research on biomimetics, bio-inspired robotics, but also a broad use of of neural networks, fuzzy logic and genetic algorithms for information studies, communications, and finance. Table 2.1 presents a brief resume with the key contributions that originated the design of robotics, and the use of bio-inspired features and mechanisms to solve engineering problems. The combination of bio-inspired features with engineering developments is advantageous in a sense of energetic efficiency and opportunities for applications of a single mechanism into solving a wide range of problems (Bekey, 1996; Habib et al., 2007).

Current trends in bio-inspired research, according to Yuan and Yang (2009) and Siciliano and Khatib (2008), are:

1. Reproducing human characteristics and behaviours (visual-perceptual capability, manipulation of objects in manufacture, robots that make facial expressions), including the human developmental process.
2. Reproducing animal vision, audition and tactile systems for robotic path planning applications (e.g., robotic whiskers), along with odour and taste recognition.
3. Developing human friendly robots that substitute animal companions.
4. Developing shape shifting robots that are able to reconfigure their morphology.

Table 2.1: Presentation of the milestones of bio-robotics.

YEAR	CONTRIBUTIONS
1920	Capek uses the word *robot* for the first time, in a novel (Norman, 2004-2011).
1937	The term *artificial intelligence* is first conceived by Turing and von Neumann (Norman, 2004-2011).
1943	A mathematical model of a neural network is proposed by McCulloch and Pitts (Norman, 2004-2011).
1948	von Neumann compares computers (automata) and the human brain. Wiener defines the principles of cybernetics (Norman, 2004-2011).
1949	von Neumann compares the function of genes to the capabilities of self-reproducing automata (Norman, 2004-2011). Turtle robots created by Grey Walter (Sharkey, 2002).
1951	Berkeley built the first electronic robot, a squirrel-like device (Norman, 2004-2011).
1958	Rosenblatt introduced the perceptron, an artificial neural network that learns by trial and error, simulating the process of human thinking (Norman, 2004-2011).
1968	First research on stability of walking robots (Garcia et al., 2007).
1972	First biped robot in Japan (Garcia et al., 2007).
1980-1990	Humanoid robots start to be built by several companies and research laboratories, Brooks starts the branch of behaviour-based robotics, and complex behaviours are derived from neural networks by Braitenberg (Sharkey, 2002). Cooperative robotics (Cao et al., 1997).
1990-2000	Robots as models of biological systems, evolutionary robotics to improve control and co-evolution, reinforcement learning for control (Sharkey, 2002). Study of insects for hexapods and cockroach-like robots, controlled by neural networks (Beer et al., 1997). Quadrupeds (Berns et al., 1999). Aggregation mechanisms in robotics (Martinoli et al., 1999). Vision acquisition and processing for guidance, navigation and maping (Franceschini et al., 1992). Micro-robots and multi-robot systems inspired by insects (Franceschini et al., 1992; Mitsumoto et al., 1995; Beer et al., 1997). Self-organising systems (Cao et al., 1997). Behaviour-based robotics (Matarić, 2001). Locomotion of humanoids through muscles and control (Garcia et al., 2007). Robot localisation and map generation (Durrant-Whyte et al., 1996).
2000-2012	Dynamic walking and running for legged robots by neural oscillators (Or, 2010). Robots for multiple-media (Bachmann et al., 2009). Emotion expression and perception with social learning (Garcia et al., 2007).

5. Applying biological mechanisms to engineering, in control systems and automation, e.g. visual dynamic tracking systems; uses of neural networks and fuzzy logic for control; planning with genetic algorithms, reinforcement learning and fuzzy logic.
6. Studying behavioural principles (brain mechanisms) and applying them to engineering, e.g. models of brain learning for grip control. These studies also include applying learning schemes (associative, reinforcement or imitation based) and evolutionary processes through simulated nervous systems for robot control.
7. Studying coordination and cooperation behaviours, and communication mechanisms, for collective robotics (swarms).
8. Incorporating real biological components into robots (biohybrid robotics), like silkmoth antennas as sensors.

The project presented in this thesis contemplates taking inspiration on animal anti-predator behaviours to create an intelligent control system for a robot. Nature as it is known today is the result of millions of years of evolution. Many processes are more energy efficient than machines, and so complex that despite decades of research, their full understanding has not been completed. Biology has inspired the development of new and better solutions compared to existing engineering ones, as many biological features can be adapted to solve different problems from other real life fields (Bar-Cohen and Breazeal, 2003). The design of a control system that is able to navigate autonomously, and also escapes from pursuers, requires a mechanism that can perform control actions towards that end.

The use of biological inspiration can occur in different areas that form part of a robotic system (Dillmann et al., 2007; Siciliano and Khatib, 2008):

1. **Design**, including the mechanics and the electronics of the actuators for locomotion that form the morphology of the robot.
2. **Control**, with a great part of inspiration coming from neurobiological research. Control can be performed on a high level or low level over a system.
3. **Perception**, considering the configuration and processing of information from sensors.

In order to illustrate the application of bio-inspiration to the areas of a robotic system, the examples given in the definitions of biomimetics and biologically inspired robots highlight the relevant areas.

2.1.1 Biomimetics

A definition of biomimetics applied to robotics is the replication of a biological mechanism or feature of an animal (mammal, insect, bird, fish, reptile, amphibian) in a robot, to gain understanding and possibly deriving a mathematical model (Bar-Cohen and Breazeal, 2003; Siciliano and Khatib, 2008). Nevertheless, an exact replication of features is not feasible, as nature relies on morphology and properties of materials to implement the

Table 2.2: Different examples of the application of biomimetics in robotic projects.

YEAR	PROJECT
2007	Implementation of a rat whisker sensory system in a robot, for testing neural models and to be used as sensors for robotics in general (Pearson et al., 2007). **Design, control and perception**
2007	Small robots that are able to walk on water by using the surface tension. The key features of the legs and actuation mechanisms of the robots have been designed according to the characteristics of the water strider insects (Song and Sitti, 2007). **Design and control**
2008	Robot that represents the chewing process of a human jaw through actuators, following the kinematics of the real movement (Xu et al., 2008). **Design, control and perception**
2008	Underwater robot that imitates the locomotion of turtles by four flippers. The designed and implemented flippers are able to make the robot move underwater and on ground. A buoyancy controller regulates the ascent and descent of the robot underwater (Low et al., 2008). **Design and control**
2008	Micro air vehicle with physical characteristics like size and motion of the wings like a Diptera insect (Wood, 2008). **Design**
2009	Flying robot that mimic the dragonfly structures of its wings. The robot has two independent pairs of wings that emulate dragonfly aerodynamic characteristics. The design of the wings is able to achieve some of the simple kinematics of a real dragonfly (DiLeo and Deng, 2009). **Design**
2009	Design and implementation of a mechanical knee that does the same functions of a human knee. The designed knee is controlled by a central pattern generator (neural oscillator) and a reflex mechanism (Renjewski et al., 2009). **Design and control**
2009	Robotic undulating-fin fish built to understand and model the wave propulsion systems for underwater applications (Liu and Jin, 2009). **Design**
2009	Robotic fish capable of three-dimensional locomotion. The mechanical design of the robot emulates a fish shape, with a flexible posterior body, flapping tail, and pectoral fins with three-dimensional movement. The control system of the robot for the movement of the joints is based on a central pattern generator (neural oscillator), after the obtention of the mathematical model of the joints (Zhao et al., 2009). **Design and control**
2009	Vision based flying insect mechanisms that have been used in robotics for negotiating narrow gaps between obstacles, collision avoidance, regulating flight speed and smooth landing (Srinivasan et al., 2009). **Perception**
2009	Model of a wing, based on the guillemot, that can be used for propulsion in air and underwater. The model, obtained from empirical simulations with an artificially built wing, is used to analyse factors involved in the evolution that led to the current functions of the guillemot wings, and the compromises in performance due to the use of the wings for two different media: air and water (Lock et al., 2010). **Design**

features (embodiment) (Pfeifer et al., 2007). Robotic implementation of biological features is only possible with available resources, but embodiment remains a challenge, as the current technology might not match the level of complexity in biological features. Research in bio-inspired morphology and materials will make other desired animal features available for robotics.

The work on biomimetics over the past decade has been extensive. Several universities around the world have dedicated research centres for the development of mechatronic devices that emulate biological systems. Some examples of projects in the area of biomimetics in the last five years are presented in chronological order in Table 2.2. The specific area of the robotic system that has been inspired by biology is highlighted in bold letters.

Mimicking nature through robotics provides alternative tools to evaluate mathematical models of biology, instead of using animals, and isolates characteristics by simplifying the system. Nevertheless, the replication of biology into robotics is not a trivial process, as technology will always limit the complexity of the implementation in terms of hardware and software. Also, building complex robotic platforms just to understand how nature can be replicated using available technology or to try models seems incomplete without diverting into the application of the research to more rewarding tasks, like solving engineering problems or for the optimisation of systems.

Reviewed literature in the area of biomimetics applied to robotics did not contemplate any examples of replication of prey-predator interactions, nor the creation of robotic platforms that would emulate any anti-predator behaviour. Nevertheless, it is not the aim of the designs in the thesis to replicate animal anti-predator behaviours, but to use them as inspiration to solve the problem of navigation and escape from pursuers.

2.1.2 Biologically inspired robotics

Complementing Subsection 2.1.1, the main objective of this subsection is the definition of biologically inspired robotics, and its exemplification through recent research. Biologically inspired robotics refers to the incorporation of features or mechanisms of animals into a robot physically, to solve engineering problems (already solved by nature) or for novel applications (Siciliano and Khatib, 2008), for example adding a vision system based on human eyes to a robotic rover. Bio-inspired robotics also refers to the extraction of the main idea behind physical, mental or chemical mechanisms of animals, to use it for the control (mathematically and algorithmically) of a robot, for example pheromones based algorithms. The extraction and implementation of the ideas can be done in different degrees, from using a feature from biology and replicating it with available resources like in a flexible human-like spine in Or (2010) and plastic tendons in Borghesan et al. (2010), to capturing the main core of an idea but the actual implementation is not similar to the real biological process at all, like aggregation behaviours inKernbach et al. (2009) and wheel-leg based robots in Bachmann et al. (2009). Implementation often depends on available resources regarding mathematical modelling, software, simulation platforms, hardware, and materials (Bar-Cohen and Breazeal, 2003).

A simple distinction between applications of biomimetics and biologically inspired robotics is difficult due to their close relation, particularly the subsequent evolution of some biomimetic projects into components of more complex biologically inspired ones; e.g., using adhesive tape that replicates the physiognomy of the feet in a gecko into robots so they are able to climb walls, in Habib et al. (2007). Nevertheless, some specific projects were originated as biologically inspired; e.g., aerodynamic structures in the shape of seeds of the tumbleweed plant for exploration in Mars, in Southard et al. (2007). Figure 2.1 illustrates the degree of mimicry or inspiration that is present in biomimetics and bio-inspired robotics, with some examples from the literature review.

Several examples of biologically inspired robotics from the last five years are presented in chronological order, according to their degree of inspiration: replicating biological features or mechanisms with available resources for other tasks in Table 2.3, or extracting the core idea of the biological feature or mechanism and implementing it in any imaginable way in Table 2.4. The specific area of the robotic system that has been inspired by biology is highlighted in bold letters.

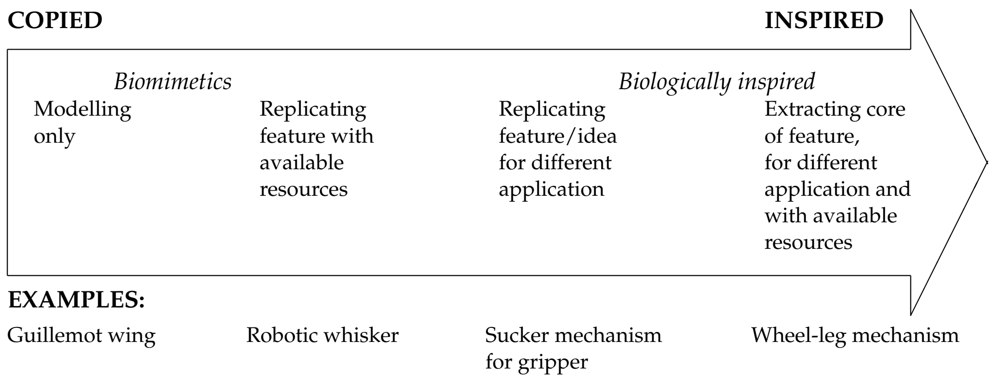

COPIED
INSPIRED
Biomimetics
Biologically inspired
Modelling only
Replicating feature with available resources
Replicating feature/idea for different application
Extracting core of feature, for different application and with available resources
EXAMPLES:
Guillemot wing
Robotic whisker
Sucker mechanism for gripper
Wheel-leg mechanism

Figure 2.1: Degrees of mimicry or inspiration in biomimetics and biologically inspired robotics.

Table 2.3: Examples of projects using biological features/ideas for other purposes.

YEAR	PROJECT
2007	Robotic fin that produces a range of motions similar to a real one, for the propulsion of underwater vehicles. The mechanism is built using servomotors (Tangorra et al., 2007). **Design**
2007	Round rovers for Mars exploration that use wind to travel great distances with little power, inspired in the tumbleweeds. The robot has fabric sails that allow controlling the movement. The design has been simulated for different kinds of wind (Southard et al., 2007). **Design and control**
2007	Quadruped robot motions controlled by a central pattern generator that is capable to climb up stairs, and find equilibrium in inclined and unstable terrains . The control system complies with generating behaviours of inclination and folding that animals with four legs use for different situations of navigation (Kimura et al., 2007). **Control**
2007	Artificial sucker mechanism for robotic manipulators, inspired by the suction in octopus tentacles. A model of the real interacting muscles in the suction process is simplified for the physical implementation with actuators (Grasso and Setlur, 2007). **Design**
2009	Chewing robot that performs different complex human-like movements to test dental materials. The robot moves according to the chewing cycle of a human mandible, creating a wear in materials (Raabe et al., 2009). **Design and control**
2009	Mechanical design of a walking robot with three legs.The design reduces the number of motors in other designs with different numbers of legs (Liang et al., 2009). **Design**
2010	Leg design with four degrees of freedom for six-legged robots. Previous proposed designs for six-legged robots have three degrees of freedom only (Roennau et al., 2010). **Design**
2010	Control of a flexible human-like spine for the walk stability of biped robots. The control is based on a hybrid central pattern generator (neural network) and zero moment point (feedback-control) control system that combines both approaches. The stability of artificial flexible spines in real-time allows the robot to adapt to dynamic environments (Or, 2010). **Control**
2011	Groundhog-like robot for mine safety exploring and rescuing, that works under hazardous conditions. The sensorial system is able to predict disasters inside the mines, and the robot can also be used to deposit sensor network nodes along its path. The robot has a basic biologically inspired shape and a moving head with a design to avoid problems in traditional excavating techniques (Zhang and Gao, 2011). **Design**

Table 2.4: Examples of projects using the extracted core of biological features/ideas.

YEAR	PROJECT
2007	Solving problems of locomotion in complex terrains, by combining jumping with rolling or gliding in robots. The jumping is achieved by storing and then releasing energy by compressing the rolling or the gliding mechanism, like animals would compress the legs (Armour et al., 2007). **Design and control**
2008	Feedback error and unsupervised learning based control system for robotic adaptive arm motions, inspired by the learning from the cerebellum and the elasticity of the muscles (Sungho, 2008). **Control**
2008	Rat brain structures based architecture for robot mapping and navigation. SLAM (simultaneous localisation and mapping) is performed extracting features from landmarks, and Hebbian learning (neural networks) to build the map. Reinforcement learning is used to learn goals of navigation (Barrera-Ramirez and Weitzenfeld-Ridel, 2008). **Control**
2009	Robot that uses a wheel-leg architecture in lunar exploration, along with light based obstacle sensing (Dunker et al., 2009). **Design**
2009	Immune response based learning mechanism for two mobile robots (learner and helper), in a track. Two algorithms were used, the behaviour arbitration (innate response: a set of rules) and the clonal selection (adaptive response: based on reproduction of suitable cells, some become a memory, others become anti- antigens). Attacks by antigens are replaced by weak areas of the track, where the learner robot may go out of its path, triggering a response from the helper robot, and storing the problematic state and solution (antibodies) for future similar cases (Chingtham and Nair, 2009). **Control**
2009	Micro air-land vehicle with a bat-like wing for flying and a wheel-leg mechanism with modified servo-motors for navigating in a terrain. The navigation is controlled from land, from the images of a built-in camera (Bachmann et al., 2009). **Design**
2009	Aggregation algorithm for mobile micro-robots based on honeybees thermotactic aggregation behaviour. Honeybees find the warmest place in an environment by executing random walks collectively. The algorithm has been implemented in a sensor-actuator architecture, measuring light instead of temperature (Kernbach et al., 2009). **Control**

The examples mentioned in Tables 2.3 and 2.4 separate the degree of inspiration within the area of biologically inspired robotics, although in some cases the line between copying a feature (mimicking) and using the feature as inspiration is not very clear. The examples in Table 2.4 demonstrate the concept of biological inspiration more clearly. Unfortunately, authors tend to use the terms *biologically inspired* and *biomimetics* indiscriminately across their publications. This whole section on bio-robotics has been written to try to define and separate the two main subareas, biomimetics and bio-inspired robotics, so the reader would be able to understand the aim and the expectations of the work presented in this thesis.

The use of inspiration from biology into machines, including robots, has evolved into a better performance, a miniaturisation of the hardware, and an increment in the intelligence (Dario et al., 2005). Considering the advantages of using bio-inspired features and mechanisms when implemented to solve engineering problems, the main backbone if this thesis is the use of biologically inspired anti-predator behaviours to achieve autonomous navigation and escape from pursuers of a mobile robot.

The solution to the problem of autonomous navigation from a biologically inspired perspective has been more oriented to the use of artificial neural networks or other types of controllers that function in a similar way to a determined species, for a specific physical model of robot in a determined media (water, air, ground). Proposed biologically inspired solutions to the problem of individual evasion from pursuers were looked for, but none was found.

The next section presents a compilation of reviewed literature regarding prey-predator interactions, including general anti-predator animal behaviours (the main biological inspiration for the designs presented in this thesis), as well as different models that have been proposed to study prey-predator interactions from different points of view (including population dynamics, individual contributions to the population dynamics, some anti-predator and predator behaviours as motion models, evolutionary studies, and computationally enhanced models for other applications.

2.2 Studies on prey-predator interactions

The first modern studies on anti-predator behaviours and prey-predator interactions date between the 1920s and the 1940s. The use of mimicry and camouflage to prevent predation, as well as mathematical models that describe prey-predator interactions are among these first studies (Sumner, 1935).

This section presents a general perspective on anti-predator behaviours and prey-predator interactions. Studies on anti-predator behaviours include predator recognition, morphological adaptations, and evasion behaviours. Prey-predator interactions have been modelled using different mathematical approaches like population dynamics, game theory, evolutionary algorithms, and others. The review on prey-predator interactions includes:

- Anti-predator behaviours from a behavioural biology perspective, with focus on individual behaviours that are feasible to implement in autonomous mobile robotics.
- A general basic review of prey-predator interactions, from a mathematical biology perspective, which refers to the study of density population distributions in population dynamics from specific models and spatial pattern formations.
- A resume of fusions of population dynamics models and game theory concepts (equilibrium, individual strategy selection, costs and payoffs). The fusion tries to incorporate the effect of individual behaviours into the prey-predator interactions and the resulting changes in the distribution of populations.
- A brief mention of food web models, another way to model the connections between species in an ecosystem.
- Other mathematical models, including motion camouflage, predator-prey chasing and fleeing respective motions, and collective motion anti-predator behaviours.
- Evolutionary algorithms that have been incorporated into population dynamics models, fusions of game theory and population dynamics models, and behaviours modelled as neural networks or other kinds of controllers, to analyse the effects of the co-evolution of species in prey-predator interactions.
- Enhanced anti-predator behaviours and implementations into robotic platforms. Some of the anti-predator behaviour models have been modified for other applications, including simulations.

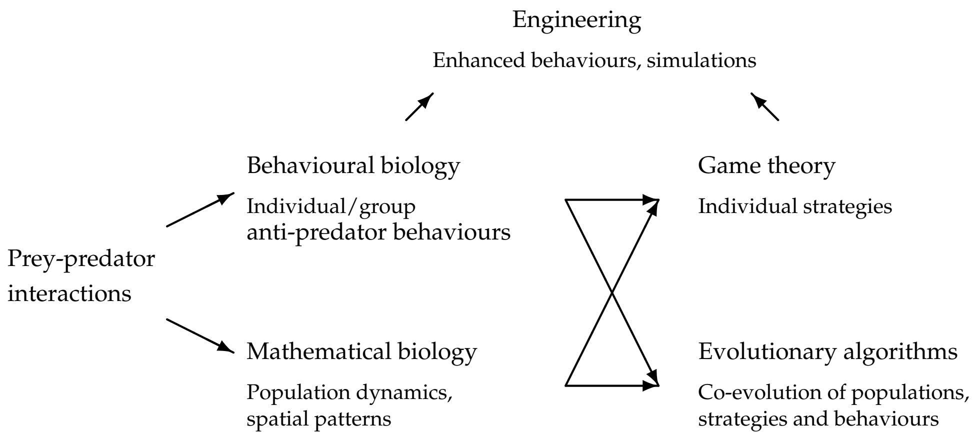

Engineering
Enhanced behaviours, simulations
Behavioural biology
Individual/group anti-predator behaviours
Game theory
Individual strategies
Prey-predator interactions
Mathematical biology
Population dynamics, spatial patterns
Evolutionary algorithms
Co-evolution of populations, strategies and behaviours

Figure 2.2: Related areas that study prey-predator interactions.

Figure 2.2 represents the relations between the areas that study prey-predator interactions.

Reviewed literature in the following subsections focuses on presenting the most relevant work for the biological inspiration basis in a general form, and summarising the most important work of non relevant but closely related areas of prey-predator modelling. Three main key points were considered in the analysis of the literature regarding relevant work, considering its applicability to robotics:

- Finding feasible methods to identify threats in the environment.
- Finding feasible anti-predator behaviours (i.e., behaviours that can be implemented in a simple autonomous mobile robot, in unknown environments and with limited sensing capabilities, such as protean fleeing and the use of refuges), with major interest on individual anti-predator behaviours, although some group evasion studies are mentioned.
- Finding mathematical navigation models of the feasible anti-predator or evasion behaviours that can be extended to the proposed design of the control system.

2.2.1 Anti-predator behaviours

The study of predation and anti-predation behaviours in animals started in the nineteenth century, focusing on animal coloration principally. In the 1950s, defence mechanisms in birds and insects were actively studied. In the 1970s, a modern approach to behavioural ecology was started, and the volume of studies for different species has increased since (Caro, 2005).

Animal species have developed different ways to deal with the threat of predation. Some are innate, the result of evolution, and others are learned and adjusted after prey-predator interactions. Anti-predator behaviours include mimicry, camouflage, fleeing, freezing, protean combinations, and fighting back, but not all of them are relevant to the project, in terms of feasibility of implementation in real life. Fleeing and proteanism are of interest for the designs presented in this thesis.

According to Caro (2005), the triggering of anti-predator behaviours occurs in stages corresponding to the interaction between a prey (or a group) and a predator (also possible in a group). The first stage corresponds to the predator recognition by observing or matching features, innate or learned. The second and subsequent stages are a set of prevention and evasion actions according to the level of threat and other factors like the morphological characteristics of the prey and the environment. Predator recognition and the main categories of prevention and evasion actions are reviewed next. Behaviours are described in a general form, i.e. not specific to any taxonomic group nor species.

Identification of predators

Recent surveys on recognition of predators reveal two mechanisms of prey in the animal kingdom to recognise a predator: innate and learned. Innate recognition is present in species where the predictability of predators is high and the variety of predating species is low, but also in young specimens with high risk after birth and limited learning before the first predation (Caro, 2005). The second mechanism is learning which animals are likely to be predators. Chemical cues and other morphological and environmental signals trigger prey responses, which differ from species to species, used for different stages of the total prey response. Learning occurs when predators are unpredictable and variable (Ferrari et al., 2008), by adding chemical cues and morphological characteristics to the set of possible predators, or from a generalised response to multiple objects and animals that becomes more specific after interactions (Caro, 2005). The main studied taxonomic groups regarding recognition of predators are mammals, birds, amphibians, and fish, and both recognition mechanisms, innate and learned, have been found on them Veen et al. (2000); Caro (2005); Mirza et al. (2006); Ferrari et al. (2008).

Chemical cues from predator odours and other injured animals from the same prey species, after suffering predation, trigger a prey response. Learning occurs when prey are exposed to chemical cues in different intensities (Mirza et al., 2006). According to Ferrari et al. (2008), the level of risk of a predator is intensified when the chemical cues are more concentrated, and the prey response is equivalently intensified. A generalisation of predator recognition happens when the prey has learned, from exposure to chemical cues, that an animal is a high level threat. The prey will react to the risk when other animals that are similar in species to the original predator are detected. If an animal is a medium level threat, the prey reacts only to that particular species (Ferrari et al., 2008).

Morphological and behavioural anti-predator behaviours

In order to avoid predation, animals have developed the following categories of anti-predator behaviours, from Caro (2005):

1. **Avoiding detection through morphological (inherent to their physical characteristics) and behavioural adaptations** (Chapters 2-3). Morphological adaptations, particularly crypsis (use of colours) allow avoiding detection, and are also known as camouflage. The first adaptation for detection avoidance is matching the background by colour resemblance, melanism (spots on the skin in leopards), seasonal coloration and masquerade (physically similar to inedible objects, like stones). Another form to avoid detection is the concealment of shadows on the body, by pigmentation or crouching. Other adaptations include disruptive coloration, the change of pattern coloration over different parts of the body to alter the silhouette of the animal, and being physically different from the animals of the same species (apostatic selection). Most of the morphological adaptations are the result of evolution.

   Behavioural mechanisms also allow avoiding detection. Animals reduce and change the times of their normal activities like foraging and reproduction, specially in response to chemical cues. Another behavioural mechanism is the building of refuges, areas with more safety, either as physical natural or built structures, or as habitat shifts.

2. **Vigilance** (Chapters 4-5). Vigilance is the action of scanning for possible predators, with the individual benefit of increasing the probability of detecting a predator. Group vigilance reduces the effort of individual vigilance, increases the time of foraging, and allows the start of other anti-predator actions faster (e.g., fleeing).

3. **Warning signals** (Chapter 6). Acoustic warning signals are used to manipulate other individuals to attract the predator towards them, to call for grouping (recruiting), to distract the predator from the offspring, and to deceive. At the same time, warning signals require energy and time, but also increase the chances of capture.

4. **Grouping** (Chapter 8). Animals form groups because the risk of predation is reduced. Grouping allows creating confusion by simultaneous proteanism of the individuals, mobbing, and saturating the predator (i.e., escape while the predator is dealing with an individual prey).

5. **Defences (also morphological and behavioural) including mobbing in groups, freezing, fleeing and last resort (attacking)** (Chapters 9-11). Defence mechanisms allow chasing the predators out, or a counterattack. Morphological and physiological defences, a result of evolution, include spines, dermal plates, weapons (claws, hooks, horns), malodour and unpalatability, venom, body size, and locomotion. Mobbing is a group defence mechanism that adds the creation of confusion, alerts other animals, and attracts a second degree predator (a predator for the current predator). Mobbing consists of approaching, observing, and harassing (following or attacking) a predator. Mammals, birds, insects and fish resort to mobbing, at the cost of energy and risk due to the closeness to the predator.

   Different morphological and environmental factors make animals escape from predators, in terms of flight initiation distance. The studied factors include temperature of the environment, the prey group size, the experience with predators, the physical

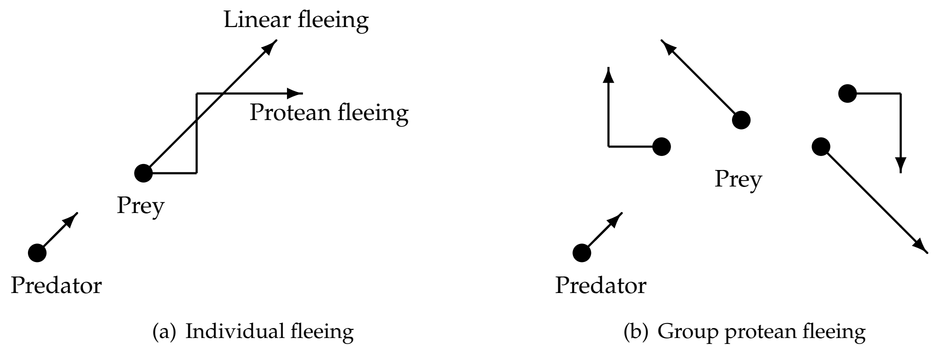


Figure 2.3: Example of an individual fleeing trajectory with and without proteanism, and example of a group protean fleeing display.

condition and morphological adaptations of the prey (size, presence of tails, agility, age), the habitat selection, the characteristics of the predator (body size, exposure to the face), the distance to a refuge, among others (Stankowich and Blumstein, 2005). The possible escape actions are flying, running away (fleeing), jumping out of the way, climbing or dropping from trees, freezing, counterattacking, and entering water. Proteanism is incorporated to increase the probability of the escape.

Particular attention has been paid to the individual defence mechanisms of fleeing and proteanism in the literature review, due to their feasibility of implementation in robotics. Humphries and Driver (1970) defined the concept of *protean behaviour* as any behaviour that is unsystematic to prevent prediction. Protean behaviours delay and reduce effectiveness in the reactions of the predator, and also work against learned predatory strategies. The protean fleeing of an individual (named *single erratic behaviour*) contemplates the use of zig-zag movements, spinning, looping or bouncing, as an emergency reaction to a predator or to prevent an attack. In groups, protean strategies include scatter displays, luring by some individuals or mobbing, and deterrence. Examples of protean fleeing, individual and in group, are shown in Figure 2.3. Although protean behaviours have been studied in specific species, no clear application to engineering problems has been noted.

The implementation of many of the morphological and anti-predator behaviours mentioned previously, is outside the scope of the project presented in this thesis. Replicating all the combinations of anti-predator categories in different stages of a predator-prey interaction into robotics requires the conjunction of several areas of knowledge, and a full understanding of generalised animal behaviours that has not been completed yet (Caro, 2005). Nevertheless, avoiding detection through refuges and individual escape actions, such as fleeing with added proteanism, can be used as an inspiration to create evasion mechanisms for a robot in the task of autonomous navigation and escape from pursuers.

Table 2.5: Highlight of the milestones of prey-predator interaction models.

YEAR	CONTRIBUTIONS
1810	Food web analysis is first used (Williams and Martinez, 2004).
1925-1926	Lokta-Volterra model, introduced for the study of prey-predator interactions based on differential equations (Morin, 1999; Berryman, 1992).
1927	Food webs and food chains are used to represent community patterns by Elton (Morin, 1999).
1935	Nicholson-Bailey model, simple model to describe a host-parasite system (Morin, 1999; Berryman, 1992).
1952	Discovery of spatial pattern formations in reaction-diffusion equations by Turing (Garvie, 2007).
1960	Leslie-Gower model, incorporating density independence for prey and predator populations (Morin, 1999; Berryman, 1992). First use of game theory in evolutionary biology by Lewontin (Sigmund, 2001).
1963	The Rosenzweig-MacArthur model, a modification of the Lokta-Volterra model in absence of predators and the rate of predation is presented (Meza et al., 2005).
1970	Parrish and Saila model, for two competitive prey species (Morin, 1999). Topological food-web analysis (Williams and Martinez, 2004).
1971-1975	Hamilton combines game theory and evolutionary biology (Sigmund, 2001).
1972	Cramer and May model, for coexistence of two competitive prey species when predators are gone (Morin, 1999).
1973-1982	Evolutionary game theory mathematical formulations and first textbook by Maynard Smith and Price (Sigmund, 2001).
1974	Reaction-diffusion equations applied to modelling prey-predator interactions by May (Garvie, 2007).
1975	Roughgarden and Feldman model, for the coexistence of a third competitive prey species (Morin, 1999).
1976	Comins and Hassell model for the coexistence of several competing prey species (Morin, 1999).
1977	Holling-Tanner model, a modification of the Leslie-Gower model with a nonlinear prey response (Morin, 1999). Lokta-Volterra model with diffusion is studied by Jorné (Satulovsky and Tomé, 1994).

2.2.2 Population dynamics, game theory and food chains

Different mathematical models of prey-predator interactions have been proposed as part of the study of population dynamics in mathematical biology, to understand the fluctuations in the species as a result of predation and many other individual, external and internal factors. Table 2.5 presents the milestones in the development of models of prey-predator interactions as populations, mentioned before. The models can be divided into the following groups:

- The study of population density distributions and pattern formations, through models based on differential equations for two or more prey populations, including the Lokta-Volterra, the Leslie-Gower, the Rosenzweig-MacArthur and the Nicholson-Bailey models, and the use of reaction-diffusion equations. Studies also include the application of evolutionary algorithms to the models, reviewed in Section 2.2.4
- The fusion of game theory concepts and density distribution models, to evaluate the process of strategy selection of individuals in prey-predator interactions, and not the populations as a whole, as in the previous group.
- Food web models, which emphasise the consumption relations between predator and prey species in an ecosystem.

The Lokta-Volterra model is described by two coupled differential equations

$$\frac{dN_1}{dt} = \alpha_1 N_1 - \lambda_1 N_1 N_2 \tag{2.1}$$

$$\frac{dN_2}{dt} = \lambda_2 N_1 N_2 - \alpha_2 N_2 \tag{2.2}$$

where N_1 is the number of individuals of the prey species, N_2 is the number of individuals in the predator species, α_1 is the grow rate of prey, α_2 is the death rate of predators, $\lambda_1 N_1 N_2$ is the decrease of the prey population due to predators, and $\lambda_2 N_1 N_2$ is the growth in the predator population due to eating prey (Bradshaw and Moseley, 1998). The other mentioned models incorporate more terms into the equations like gradients of diffusion of predators and prey (Guan et al., 2011).

The models concerning population density distributions have been analysed from different perspectives (Morin, 1999):

- Finding oscillatory solutions to the equations, representing natural cycles of predation, death and regeneration of populations, and stability of cycles, i.e. the presence of stable limit cycles. For example, Bradshaw and Moseley (1998) studied oscillations in populations using the Lokta-Volterra model; Guan et al. (2011) constructed a Lyapunov function to analyse the stability of the Leslie-Gower model; Gakkhar and Naji (2003) studied the stability of a Holling-Tanner model.
- The generation of spatial patterns in the distributions of populations. Among the examples, Petrovskii and Malchow (1999) studied the formation of irregular patterns using a reaction-diffusion equation; Liu and Jin (2009) analysed the formation of spatial patterns in a Lokta-Volterra model; Neubert et al. (1995) used integro-difference equations to study spatial pattern distributions and dispersal functions.
- Impact of external and internal factors (aggregation, refuges, illness, available resources, sampling, intraguild predation) in the population densities and pattern formation. Examples include the application of the Lokta-Volterra model to groups and solitary individuals differently to study effects of cooperation and aggregation in population densities by Mchich et al. (2006); Guan et al. (2011) concluded that refuges influence strongly the formation of patterns; Meza et al. (2005) used a control theory approach to analyse the stability of a Rosenzweig-MacArthur model considering exploitation of resources and threshold policies for the populations; and Ives et al. (2005) considered different factors related to the availability of consuming resources that affect populations, such as sampling (survival and reproduction of some individuals), resource partitioning within species, and intraguild interactions (killing potential competitors), in a Lokta-Volterra model for multiple species.
- Analysis of the resulting dynamics after applying evolutionary algorithms to the models. For example, Dercole et al. (2006) evolved a Rosenzweig-MacArthur model to analyse the stability of the ecological cycles in the populations.
- Optimisation of computation of solutions. Examples include the use of finite difference methods to solve the reaction-diffusion equation in Garvie (2007).

Population game theory combines the dynamics of animal behaviour with the study of population distributions, also known as population dynamics (Cressman et al., 2004), to find equilibrium points (stability in the spatial distributions) from a more individual perspective. Cressman et al. (2004) modelled an individual payoff in terms of a density population distribution model (Lokta-Volterra), as the survival and fitness of an individual depends on the distributions of other animals, and the strategy dynamics were defined using replicator equations. In Tschirhart (2004), the payoff is defined in terms of the expenditure of energy (in predation) and biomass necessary to replace it (eating prey). Population densities are analysed and equilibrium points are computed in terms of maximisation of energy. Brown et al. (1999) considered a prey-predator system as a foraging game where the predators look for maximising their success on predation and prey look for an optimal level of vigilance, using a Rosenzweig-MacArthur model of population dynamics with incorporated prey fear dynamics.

Food web analysis has been oriented to four main areas (Williams and Martinez, 2004). The first area is a topological approach, or the construction of network structures. The second area is a statistical analysis of the connections within the web and their dependence. The third area is the emergence of patterns in the distribution of connections (e.g. the presence of omnivores and their relation to stability of the community in Morin (1999)). The fourth concerns the mechanisms that take place in the food webs, by using mathematical tools, such as a bioenergetic model of dynamics that studies the stability of density population distributions as a result of energy consumption and recovery through feeding. Other attempts bring together population game theory and a bioenergetic analysis (Tschirhart, 2004).

One of the applications of the study of predator-prey interactions in terms of density of population is pest control, where a predator species is introduced to control the pest population (Ives et al., 2005). Recent research includes the interactions between multi-species predator-prey systems.

The study of density population distributions as a result of predator-prey interactions, their extensions to individuals through game theory and food chains (or food webs) are extensive research fields. This subsection only presents a small glance into the contents of research on population dynamics, as it forms part of the general context regarding the available models of prey-predator interactions, although the object of study differs from models that could be directly applicable for navigation strategies to escape from pursuers.

2.2.3 Other models of prey-predator interactions

Mathematical models of specific anti-predator collective and individual behaviours have not been developed extensively. Some interesting models that are included in this subsections are motion camouflage, chasing and fleeing motions, and individual strategy selection based on start of reaction times. Other group anti-predator models are mentioned in general, although they are outside the scope of the thesis, dealing with a single robot in the task of escape.

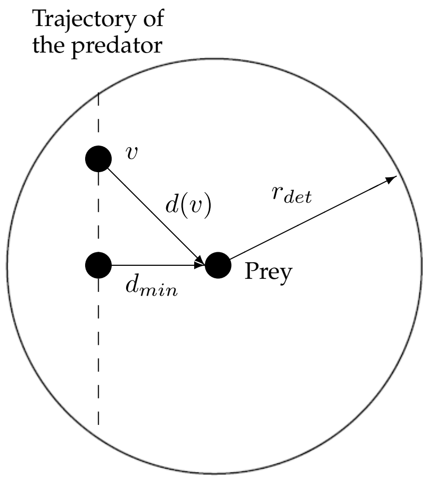


Figure 2.4: Graphical representation of the model presented in Broom and Ruxton (2005) for a prey-predator interaction.

A game to predict optimal behaviour in an individual prey-predator interaction in terms of the initial time of an anti-predator response or an attack is proposed by Broom and Ruxton (2005). Strategies for the prey and the predator can be computed simultaneously using the following assumptions: when the prey starts running before the predator does, it will not be captured; the decisions of the prey are affected by the cost of fleeing (less feeding); the prey is assumed stationary and cryptic in the beginning of the game; prey can detect predator at greater distance than the predator detects the prey; the payoff of the prey is the probability of surviving, reduced by the costs of fleeing. The model shows that the optimal strategy of a prey is running immediately after a predator is detected, or running immediately after being detected by a predator. The optimal strategy of a predator is attacking immediately after discovering a prey, or attacking when the distance to the prey is minimal if the prey has been discovered before and the distance to it is large.

The model in Broom and Ruxton (2005) is represented visually in Figure 2.4, for a predator with a linear trajectory entering the detection space of the prey with radius r_{det}. v is a point in the trajectory of the predator, and $d(v)$ is the distance from the predator to the prey, with a minimum of d_{min}. The predator can initiate an attack from any point v in its trajectory, or wait until the distance is d_{min}. The probability of detection of the prey tends to zero if the predator and the prey are far from each other, and not running is preferable for the prey to running (as it has energetic costs).

The model in Broom and Ruxton (2005) is not directly useful for the solution of the problem of navigation and escape from pursuers as it is in terms of starting the actions of fleeing or attacking, and the problem described in this thesis assumes that pursuers should start pursuing or planning and executing an interception as soon as they detect the robot, and the robot should start fleeing as soon as it detects pursuers. Nevertheless, it has inspired the definition of some of the assumptions in the designs.

Anderson and McOwan (2003), Glendinning (2004) and Justh and Krishnaprasad (2006) obtained and analysed models of the capture of prey, denominated *motion-camouflage*, where the predator moves towards the prey following a special trajectory, to make the prey think the predator is stationary. Motion-camouflage achieves reaching the prey in all circumstances, compared to *classic pursuit*, and if the predator is faster than the prey, motion-camouflage performs faster than classic pursuit. The proposed models assume a predictable linear prey trajectory while the predator has not been discovered.

The models in Anderson and McOwan (2003) and Glendinning (2004) deal with the problem of stealthy predation, therefore it is not applicable to the solution of the problem of this thesis. But it has inspired the need of intelligent evasion as a response to intelligent pursuit.

Furuichi (2002) presented a mathematical model of the motion of a prey and a predator. The prey and the predator are modelled to have head directions, and viscous resistance is also added to the model. The distance between predator and prey is converted into a repulsion or attraction force, which causes an appropriate response (chase in the case of the predator, or evasion in the form of straight fleeing or protean zig-zag in the case of the prey) according to their closeness. The model tries to replicate both predator and prey navigation schemes during the pursuit of the prey.

The predator perceives an attractive force towards the prey in terms of the distance to the prey, $|\vec{r}|$, defined as

$$\vec{F}(\vec{r}) = \begin{cases} 0 & D_1 \leq |\vec{r}| \\ B_1 \cdot \vec{r}/|\vec{r}| & D_2 \leq |\vec{r}| < D_1 \\ B_2 \cdot \vec{r}/|\vec{r}| & |\vec{r}| < D_2 \end{cases} \tag{2.3}$$

where B_1 and B_2 are constant values, and $[D_2, D_1]$ is an interval of distances where the predator can chase the prey. The prey perceives a repulsion force away from the predator, also in terms of the distance to the predator, $|\vec{r}|$, defined as

$$\vec{f}(\vec{r}) = \begin{cases} 0 & d_{sight} \leq |\vec{r}| \\ b_r \cdot \vec{r}/|\vec{r}| & d_s \leq |\vec{r}| < d_{sight} \\ b \cdot \vec{e} & |\vec{r}| < d_s \end{cases} \tag{2.4}$$

where b_r is a constant for the force, d_{sight} is the radius of detection of the predator by the prey, d_s is a smaller distance where the prey shifts from a linear repulsion movement in the same direction of the vector between the predator and the prey to a zig zag with a random direction given by $\vec{e}$, changing at random time intervals. Figure 2.5 represents the model by Furuichi (2002) visually.

The most relevant model to the work presented in this thesis, based on individual actions to avoid predation, is the one in Furuichi (2002), considering the use of protean navigation and fleeing as anti-predator responses. Fleeing with added proteanism is presented as a response to the chase motion, in a more deliberative manner than repulsive and attractive forces in Furuichi (2002). The use of forces to model the interaction has inspired the implementation of a planning approach with artificial potential functions, which incorporate obstacle avoidance and information to make predictions about the future to compute a

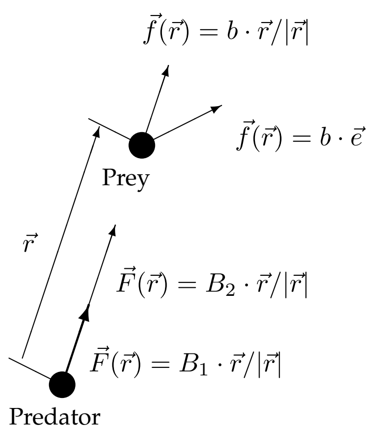

$\vec{f}(\vec{r}) = b \cdot \vec{r}/|\vec{r}|$
$\vec{f}(\vec{r}) = b \cdot \vec{e}$
Prey
$\vec{r}$
$\vec{F}(\vec{r}) = B_2 \cdot \vec{r}/|\vec{r}|$
$\vec{F}(\vec{r}) = B_1 \cdot \vec{r}/|\vec{r}|$
Predator

Figure 2.5: Attraction and repulsion forces in the model presented in Furuichi (2002) for a prey-predator interaction.

navigation path of escape.

Prey swarming causes confusion for some predators. Thus their attacks become less successful (Jeschke and Tollrian, 2007). Other proposed models involve group anti-predator behaviours, for example a fish school evading predators in Zheng et al. (2005), where three behaviour patterns for evasion are considered: schooling (as imitating other fish and collision avoidance motions), cooperative escape, and selfish escape, depending on the distance to the predator. Zheng et al. (2005) proposes a model of the predator. The behaviour patterns are assessed in terms of individual consumed energy, frequency of collisions, unification of the group, and polarisation of collective motion. Lee (2008) extended the work by Zheng et al. (2005) to the area of robotic swarms, where other elements of artificial intelligence have been added to improve the original model and to implement it on robotic platforms. This category of group anti-predator models does not focus on the development of individual anti-predator behaviours, but on the relations between elements of the group and their resulting joint behaviours.

Collectivity is useful in terms of robustness and flexibility, as it has been demonstrated in multi-robot systems for the most diverse applications. Thus, cooperation and many other biological group behaviours have been studied and extended into robotics and computer simulations, e.g. robotic swarms, and multi-agent systems. Some examples of developed artificially enhanced anti-predator behaviours, i.e. bio-inspired anti-predator behaviours that are improved or modified for other purposes, are mentioned in Subsection 2.2.5.

2.2.4 Evolutionary prey-predator interactions

Genetic algorithms have been used in prey-predator individual models (e.g., artificial neural network based navigation controllers), and game models, to study the effects of evolution on prey-predator interactions. In this subsection, the use of evolutionary studies applied to population dynamics models, reviewed in the previous section, is briefly mentioned. Then, a general overview on distinguished analytic work in evolutionary game theory is presented, with emphasis on the generation of anti-predator strategies by the prey. Finally, key studies on the co-evolution of predator and prey behaviours are presented. Evolutionary biology and evolutionary game theory complement the studies on prey-predator interactions, and give support for the use of some anti-predator behaviours like proteanism for autonomous mobile robotics, although an evolutionary approach has not been considered for the proposed designs in this thesis.

Evolution applied to prey-predator population density distribution models has been studied, to analyse the effect of co-evolution in the dynamics and stability of the populations, including other external and internal factors such as the geography of the habitat, the homogeneity of a population (Abrams, 2000). Most of the studies on evolutionary prey-predator interactions assume a co-evolution (simultaneous evolution) of prey and predator species, although individual evolution may cause instability of the system, particularly in the case of an evolved prey.

Other studies combine population dynamics models with evolutionary algorithms to study the dynamics of diversity in a food web (e.g., Drossel et al. (2001)). The co-evolution of multiple species and the effect in their populations are studied in macro-evolutionary models based on the relations of a food web in a determined ecosystem.

From another perspective, evolutionary game theory proves that unpredictability (protean behaviours) is developed as a counter-strategy against predation (Miller and Cliff, 1994; Miller, 1997). According to Miller and Cliff (1994), studies on behavioural biology have proved that humans and animals posses a *Machiavellian intelligence*, translated into the actions of *randomness* (or unpredictability), *concealment* (understood as hiding information) and *deception* (or giving false information).

Other examples of studies on evolutionary game theory include Haynes and Sen (1996), using a predator-prey (pursuit-evasion) game to develop coordination strategies for the predators from evolution. Linear prey strategies (i.e. the prey moves in a line) are effective as they avoid staying in a specific local neighbourhood. Wahde and Nordahl (1998a) analysed the evolution of pursuit-evasion games where the speed of evaders is different from pursuers. In Wahde and Nordahl (1998a), evaders were designed to be controlled by artificial neural networks, whereas pursuers of two kinds were analysed: adaptive neural network and fixed strategy based; evaders were found to develop complex protean strategies. Finally, Mitchell (2009) investigated the evolution of prey-predator games with multiple behaviours (extracted from studies on pre-encounter and post-encounter games, such as habitat selection, or waiting for a prey that is in a refuge), concluding that prey movements depend on predators learning prey locations; also, prey vigilance of predators enhances their chance of surviving.

From a behavioural perspective, Floreano and Nolfi (1997) studied the effect of adaptive protean behaviours, in the co-evolution of prey and predators, using robots to simulate prey and predators strategies, controlled by simple perceptrons. Fixed and adaptive protean perceptrons were evolved and compared, and both evolutions generated noisy controllers for the prey, illustrating the usefulness of proteanism as a prey strategy for evasion. Floreano and Nolfi (1997) concluded that the co-evolution may lead to the development of increasingly more complex behaviours (arms races), or cycling problems where effective strategies are lost and replaced by others that are better for a specific individual in a specific time, but are not effective in the long run.

Studying the co-evolution of strategies enhances their extension to artificial evolution, to develop robots that act faster and robustly to dynamic environments, by the mutual challenges that the co-evolution brings to the competitors representing prey and predators (Nolfi and Floreano, 1999). For example, competition and cooperation behaviours have been achieved from co-evolving soccer-playing robots against another team, or among their own teammates (Uchibe and Asada, 2006).

Evolutionary computational studies show that developing specific complex protean anti-predator behaviours is the result of interactions between populations of prey and predators. When designing autonomous robotic systems, it would not be necessary to develop adaptations of anti-predator behaviours (including evasion) from zero, as it happens when randomly initialised controllers are evolved using genetic algorithms. Specific anti-predator behaviours could be implemented as part of the autonomous control system, to be triggered when circumstances require them.

2.2.5 Enhanced anti-predator behaviours and applications

Bio-inspired robotics and control systems have extended biological group behaviours to the fields of formations, swarms, multi-robot systems and multiple player games as multi-agent systems (Cao et al., 1997). Anti-predator behaviours of fish schooling and extended to crowd dynamics have been implemented in simulations. Prey-predator interactions have been morphed to pursuit-evasion games and solved by different game theory and reinforcement learning techniques, reviewed in Chapter 6. Some examples of designed or artificially improved anti-predator behaviours inspired from nature are presented in the following paragraphs.

Oboshi et al. (2002) presented an evolutionary method to develop group evasion behaviours against predation. In this article, a fish chooses to move using any of four basic behaviours: attraction, repulsion, parallel moving, and searching, according to distances to other fish. For evasion, the fish choose any of the basic behaviours considering angles between them and other fish and predators. The parameters that determine which basic behaviour to use were evolved with genetic algorithms until they converged. Obtained group navigation movements are similar to real ones.

Morihiro et al. (2007) studied agent based self-organised grouping and anti-predator behaviours computed from a reinforcement learning algorithm. Agents (individuals) are attracted or repulsed by other agents, and repulsed by predators, according to a predefined reward function in terms of the distances between the elements.

Lee (2008) presented a model for group evasion behaviour used to simulate human crowds, based on biologically inspired schooling evasion patterns (from the models used by Oboshi et al. (2002) and Zheng et al. (2005)) and human sociological factors. The biologically inspired elements of the model target an evasion from imminent danger, and the sociological elements target distant danger that can be predicted.

Swarm robotics (multi-robot systems) have focused on the task of pursuit rather than evasion, principally when emulating pursuit-evasion games. Individual or group anti-predator behaviours extended from a biological model are rarely implemented on robots, and few attempts using simple group anti-predator behaviours (e.g., dividing the group and moving in two different directions) remain as simulations.

The study of pursuit-evasion games in the game theory field involve the use of extended models from prey-predator biological interactions. Although the purpose of studying the games and solving them through different methods, some from artificial intelligence, is not necessarily related to biology (as shown in Chapter 6). Moreover, the focus of pursuit-evasion games is normally the pursuing (individual or collective) of a simple evader. The definition, different forms that have been proposed to find a solution, and robotic implementations of pursuit-evasion games are reviewed with more detail in subsequent chapters.

2.3 Discussion

The design of a deliberative high-level control system that decides on an evasion navigation strategy is inspired by the idea of animals possessing a Machiavellian intelligence, trying to confuse an intelligent predator by the use of anti-predator behaviours. The goal is not to emulate animal nor human reasoning, but to take inspiration from them to construct a solution to a problem in the area of robotics. According to the definitions of biomimetics and biologically inspired robotics, it can be deduced that the use of biological inspiration from anti-predator animal behaviours to solve the problem of autonomous navigation and escape from pursuers for a robot falls into the category of biologically inspired robotics. More specifically, the area of the system in which the bio-inspiration occurs belongs to the control system, considering the division presented by Dillmann et al. (2007).

Animal species present several complex anti-predator behaviours, some of them characteristic of a taxonomical group. Morphological adaptations are the most difficult to implement in a robotic system, as they depend strongly on the available technology for the embodiment. Nevertheless, three general anti-predator behaviours are of particular interest for the solution of the problem of this thesis: fleeing, hiding and protean, as they are more feasible to use as inspiration for implementing control strategies in a robotic platform. Many species use the three behaviours to try to avoid predation, as Caro (2005) mentions. Other features that are of interest for inspiring the designs are: assessing how quickly to deliberate an evasion strategy by distance to predators and/or distance to hideaways, inspired by Broom and Ruxton (2005), and using protean behaviours in the presence of a pursuer, to try to deceive it in a Machiavellian way (inspired by Miller and Cliff (1994); Miller (1997)).

Combinations of the distances between a robot (prey) and predators or pursuers, and between the robot and possible hideaways (also named refuges in biology), can be used in a relatively simple autonomous robotic system to determine how immediate an evasion strategy is needed, inspired by the ideas of Broom and Ruxton (2005). Other features that animals normally consider when they start an evasion, according to Stankowich and Blumstein (2005), would need a complex interface of sensors and processing of signals and images that analyse the environment if implemented into a robotic system; other factors would be irrelevant if predators or pursuers are not animals.

Unfortunately, many of the mathematical models that deal with predator-prey interactions are more focused on obtaining good models for population dynamics in terms of density distributions, pattern formations and relations in an ecosystem. Although some studies incorporate individual behaviours into the models in the form of game theory based computation of strategies, few could be used to solve the problem of navigation and escape without being modified in essence, as their main focus is the analysis of population dynamics, and not deriving improved control strategies for other applications. On the other hand, evolutionary studies also based on prey-predator interactions corroborate the emerging of complex behaviours, such as proteanism, when co-evolving prey and predator together, thus supporting the use of proteanism and *intelligent* escape to solve the problem studied in this thesis.

From the few models that deal specifically with the individual dynamics in a prey-predator interaction independently from each other, such as the study of chasing and fleeing motions by Furuichi (2002), some ideas have been extracted and used to inspire a solution to the problem of navigation and escape. The attraction and repulsion forces that drive the fleeing in Furuichi (2002) and the incorporation of proteanism are particular features used as inspiration.

Although some group evasion movements have been modelled in swarm computation and robotics, it is not trivial trying to extrapolate their results into a single individual. The problem of escape in a single robot is limited to the capabilities of the robot to process the environment and to decide what to do in that specific situation, without the help of any other individual, and at a greater cost if it is unsuccessful in the escape.

Prey behaviours and their implementations into robotic applications have been mostly ignored, but no reason has been given for that. Are they inefficient in terms of energy consumed? Are prey anti-predator behaviours so disadvantageous or too complicated to implement compared to predation ones? And why do most of defence, surveillance and rescue applications base their developments in search and destroy, capture or find, respectively (i.e., developing intelligent pursuers)? No clear answer to the previous questions is to be found in the literature. But other researchers agree that robots that are designed to exist in the most common habitats, surrounded by pets, wild animals, children, people, or any other danger, need to deal with the possibility of being chased, as real beings do in nature, hence the need to design robots that can deal with the most basic tasks of survival and self-protection emerges, if full autonomy is to be achieved someday. Furthermore, the problem of escape from pursuers becomes more complex when adding unknown, dynamic environments, and limited sensing capabilities.

The extraction of features from fleeing, hiding and protean anti-predator behaviours as inspirations from biology to solve the problem of navigation and escape from pursuers from a path planning and game theory perspective are detailed in Chapters 4, 5, and 7.

2.4 Concluding remarks

This chapter presented a brief literature review on the area of bio-robotics, adding some recent examples of research projects to illustrate the difference between biomimetics and biologically inspired robotics. The definitions were used later to set into context the work presented in this thesis. The project in this thesis contemplates the use of anti-predator behaviours as an inspiration to solve the problem of autonomous navigation and escape, implemented through path planning and reinforcement learning algorithms.

Subsequently, the chapter presented a literature review on prey-predator systems from different points of view:

- Biology and ecology: general anti-predator behaviours of different taxonomic groups.
- Mathematical biology: a review on models for prey-predator interactions, which deal with population dynamics, pattern formations, properties of the models, interference of different factors (internal and external) in the dynamics, effect of individual actions (through the use of game theory concepts) on the dynamics, and interrelation of species in ecosystems (food webs).
- Evolutionary biology: a brief resume of the trends in the studies on the co-evolution of prey-predator interactions. Some studies demonstrate the emerging of complex behaviours as result of the evolution.
- Other models of anti-predator behaviours and prey-predator interactions: three interesting models of individual behaviours of chasing and fleeing, motion camouflage and attack and reaction, and examples of group anti-predator behaviour models.
- Artificial improvements of anti-predator behaviours: brief review on the uses of biology concepts and models to create improved technological systems, such as simulation tools.

The review of the different models and studies on anti-predator behaviours and prey-predator interactions highlighted the lack of models of individual real actions and reactions of different animal species. None of the models found in the literature can be used directly to solve the problem of collision avoidance and escape from pursuit for a mobile robot with limited sensing capabilities. Nevertheless, the models by Furuichi (2002), Broom and Ruxton (2005) and the evolutionary studies in proteanism can be used to construct such a system that performs individual actions equivalent to some anti-predator behaviours for the robot.

Finally, a discussion of the presented literature review takes place, highlighting the gaps on the use of anti-predator behaviours as inspiration to improve current technology, and which concepts in specific literature have inspired the designs contained in this thesis.

The next chapter presents a comprehensive literature review on general path planning methods, with main focus on potential functions. A deeper review on previous work related to potential function implementations and challenges is provided, with some examples and explanations added to highlight the functionality of the different proposed models. The concepts provided in this chapter and the following serve as the basis for the designs presented in Chapter 4.

Chapter 3

Autonomous navigation with potential functions

The study and design of robots that are capable of successfully navigating autonomously in challenging environments that involve pursuit and possible danger are advantageous and to some extent necessary for the following reasons:

- Intelligent escaping robots provide a complex and challenging platform to evaluate the success of the design of pursuers for tasks like surveillance and target interception in real life situations.
- The incorporation of escape behaviours, to avoid *predation* (also interpreted as danger or capture) adds realism to robots that interact with humans and animals, as they may encounter pursuit situations.
- Using biological inspiration for the navigation and escape from pursuers opens the door to more research into assessing the feasibility of implementing anti-predator behaviours in simulations and robotics.

The problem of autonomous navigation and escape from pursuers can be potentially solved using any of the numerous possibilities that path planning offers, or even using tools from game theory and artificial intelligence. The navigation and escape problem has been analysed from two different perspectives in this thesis:

1. As a path planning problem, where an optimal trajectory of navigation is found according to the characteristics of the environment, different kinds of constraints, and a possible goal.
2. As a pursuit-evasion game, where the opponents are the pursuers (traditionally the evaders are the opponents), and a navigation strategy is computed by maximising the pay-off of selection actions within the strategy, according to the state of the environment and the respective actions of the pursuers.

This chapter presents a review of the literature relevant to path planning methods, with an emphasis on potential functions. Pursuit-evasion games are reviewed in Chapter 6.

Section 3.1 presents a general overview of the different methods available in path planning for the computation of a navigation trajectory. Planning for autonomous navigation has been studied widely, and several algorithms have been proposed to find a path using different information from the environment in the form of a map, and combining the use of search methods. The reviewed methods include: bug algorithms (Subsection 3.1.1), roadmaps (Subsection 3.1.2), cell decomposition (Subsection 3.1.3), sample-based methods (Subsection 3.1.4), and potential functions (Subsection 3.1.5). Some of the main search algorithms used to compute an optimal or suboptimal solution for the different path planning methods are mentioned in Subsection 3.1.6. Subsection 3.1.7 summarises the main forms used by the different path planning methods to represent the environments. Section 3.2 includes some relevant work in the area of path planning for stealth or covert navigation, as it is related to hiding behind obstacles, and also reviews previous work on path planning for evasion from pursuers.

Solutions to path planning using potential functions are of particular interest, due to the advantages of this method for the solution of the problem presented in this thesis. Section 3.3 expands Subsection 3.1.5 into a comprehensive review of the main branches of research in the field of potential functions: scalar potential functions, numerical solutions to the Laplace's equation, and direct vector field from a set of functions for the obtention of a control law. The section includes a brief summary of the main challenges when planning for dynamic, sensed, previously unknown environments, and how they have been addressed by potential function based path planning techniques (Subsection 3.3.4). Also, discrete approaches to potential functions are mentioned (Subsection 3.3.5), followed by the main challenges in the implementation of the functions (local minima) (Subsection 3.3.6). The end of Section 3.3 presents a brief summary of combinations of potential functions and other path planning methods (Subsection 3.3.7).

Finally, a discussion of the methods and algorithms is included in Section 3.4. The discussion explains the selection of potential functions over other kinds of path planning methods for the solution of the problem of autonomous navigation and escape. The existent formulations of potential functions (scalar and vector based) are also analysed towards their application to represent dynamic partially observable environments with obstacles and pursuers. The conclusions of the chapter are presented in Section 3.5.

3.1 Path planning for navigation of autonomous robots

Motion planning, including path planning, is a widely studied area of research. A motion planning problem is defined as obtaining a minimum-cost feasible path that connects an initial and a final state (or configuration). In real life, a path is a set of translations and rotations in the free space, considering some constraints according to the physical characteristics and dynamics of the robot (LaValle, 2006). The term *path planning* corresponds to the calculation of kinematic variables like position or speed, while the term *trajectory planning* includes kinematics and dynamics (forces), mostly applicable to robot manipulators. Path planning can be static or dynamic, continuous or discrete, offline planning in advance for the whole navigation from an initial point to a final or online from sensors as a repeated loop of sensing, planning and acting (Choset et al., 2005).

The main approaches to solve the problem of path planning can be grouped into the following categories, according to Hwang and Ahuja (1992a); Bell (2005); Choset et al. (2005); LaValle (2006):

- Sensor-based bug algorithms
- Roadmaps
- Cell decompositions
- Sample-based methods
- Potential functions

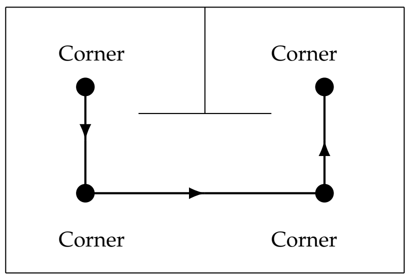
Exact and heuristic algorithms can be used in any of the approaches mentioned before. Exact algorithms find an optimal solution if it exists, or find that no solution exists. Heuristic algorithms find fast solutions, but at the cost of reliability and optimality (Al-Sultan and Aliyu, 1996). The main characteristics of each category are briefly explained in the next subsections, mentioning the advantages and disadvantages for their use in solving a problem of autonomous navigation and escape from pursuers in a dynamic environment with limitations of processing time and information. Particular interest has been given to potential fields, detailed deeper in Section 3.3.

Some of the challenges that have been addressed in path planning methods (Bell, 2005) are:

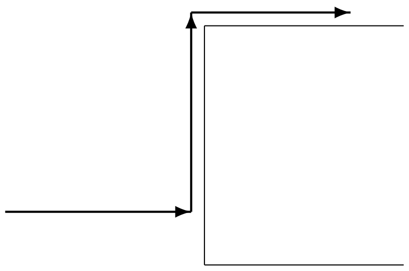
- Dealing with dynamic and complex environments
- Planning with imperfect information from the environment
- Planning for imperfect localisation of objects in the environment
- Dealing with error tolerance

Nevertheless, these challenges are still being addressed in current research. In order to design a planning system or algorithm to solve the problem of autonomous navigation and escape, considering challenges such as imperfect information, the following steps are required according to Hwang and Ahuja (1992a):

1. Obtaining a representation of the robot and the environment, i.e., the identification of the free-space and the obstacles. The representation could be done using cells in a grid, identification of landmarks, curves that join points of the free-space, or potential functions.
2. Selecting and implementing a search method (e.g., A*), or algorithm (e.g., gradient descent, following walls) to find a path as the solution, according to the optimality required by the task, and the available computational resources.



(a) Path connecting landmarks



(b) Path following geometry of obstacles

Figure 3.1: Examples of a a path connecting landmarks and a path following the contour of an obstacle, by means of bug algorithms.

3. Optimising the path locally by smoothing it according to physical constraints. This step would facilitate the introduction of a low level controller to physically move the robot to the required locations.

The search methods and the representation of the environment are not path planning methods alone, but components that can be exchanged to produce methods within the categories mentioned in Section 3.1. The procedure proposed by Hwang and Ahuja (1992a) is considered in Chapter 4 for the design of a bio-inspired methodology for the environment analysis, and for computing a path. The following subsections summarise the main five options of methods available for path planning.

3.1.1 Sensor-based bug algorithms

The simplest path planning methods are also named *bug algorithms* in some literature. The algorithms can do two main kinds of motions: linear movements towards the goal or around obstacles that appear between a robot and a goal, according to the information from the sensors (Bell, 2005; Choset et al., 2005). These algorithms are based on landmark and geometrical representations of the environment, reviewed in Subsection 3.1.7. Figure 3.1 illustrates two bug algorithms based on landmarks and following the geometry of the obstacles.

Compared to other kinds of path planning methods, bug algorithms are easier to understand and implement computationally, therefore they are recommended for robotic tasks that require simpler navigation strategies that follow landmarks. Also, their sensor-based approach allows them to deal with local information and dynamic environments. Nevertheless, bug algorithms are limited to direct navigation movements towards the goal, or circumnavigating obstacles, and some require the labelling of the landmarks.

3.1.2 Roadmaps

In path planning, a map is a data structure that contains a model of the environment (Choset et al., 2005). Maps can be topological (based on landmarks), geometric (based

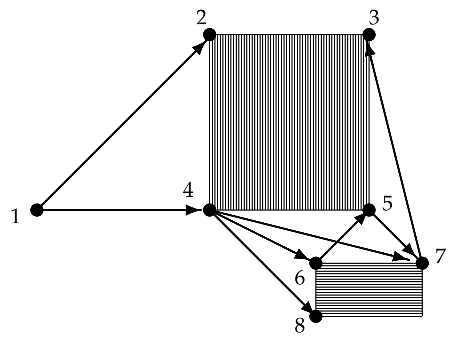


Figure 3.2: Roadmap with 8 nodes in the free-space, for an environment with two rectangular obstacles. A path would connect nodes 1and 4, 4 and 7 and 7 and 3, to go from locations 1 to 3.

on shapes or line segments) or represented by grids. A roadmap is a topological map, consisting of curves or skeletons connecting points in the free space (i.e., the space that is not occupied by obstacles). A path is formed by the union of curves from an initial position towards a goal location (Hwang and Ahuja, 1992a; Bell, 2005; Choset et al., 2005; LaValle, 2006; Siciliano and Khatib, 2008). Search methods like depth-first, breadth-first, best-first, and A* are used to compute the connections between the curves.

Roadmaps can be divided into visibility graphs (lines connecting the vertices of polygonal objects within the sight of each other), Voronoi diagrams or deformation retractions (sets of points equidistant to closest obstacles), silhouettes (lines called *slices* that connect critical points and boundaries of obstacles and the environment), subgoal networks and freeway nets. An example of visibility graphs is shown in Figure 3.2, with 8 nodes connecting the lines of vision according to the geometry of the square obstacles.

Roadmaps have the ability to deal with complex geometries in the environment, like polygons, but if the robot does not know the environment before the planning, a map built through sensors needs to be computed and stored. Also, finding closed paths that connect an initial and a goal configuration is not possible due to the loss of connection between the curves in the roadmap; this problem can be solved by incorporating different computations of the curves (e.g., hierarchical Voronoi diagrams), but at a higher computational cost. The main drawback of roadmaps is the overall high computational cost of the curves in the free space, particularly when the environment is complex. Probabilistic roadmaps have been incorporated within the group of sample-based path planning methods (Subsection 3.1.4), as the algorithms incorporate a sampling of the possible configurations.

3.1.3 Cell decomposition methods

In cell decomposition, the environment is divided into regions (cells), and an *adjacency graph* is computed to indicate cells sharing common boundaries. A path is computed from the adjacency graph to indicate movement from cell to cell, but no trajectory (i.e. specific kinematics to move towards a location) is computed (Hwang and Ahuja, 1992a; Bell, 2005; Choset et al., 2005; LaValle, 2006). Cell decomposition methods are frequently used for

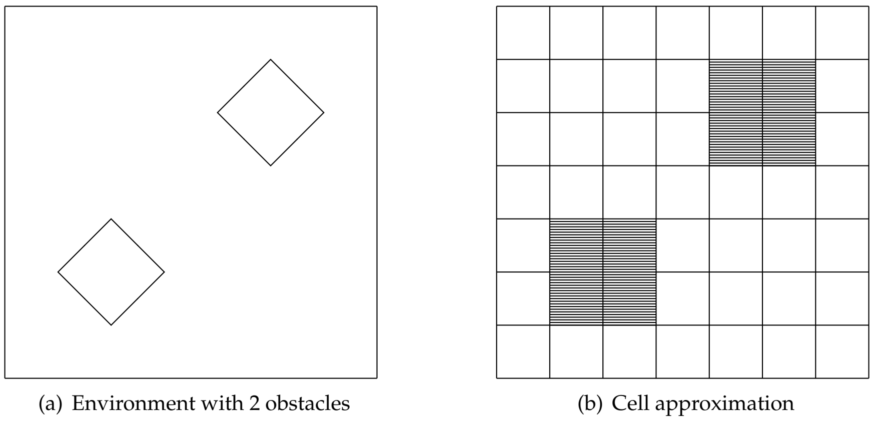

(a) Environment with 2 obstacles (b) Cell approximation

Figure 3.3: Example of an environment with two obstacles and its cell approximation, where the black cells represent the obstacles.

coverage tasks.

The decomposition of the environment into cells can be exact (using cells of different shapes to represent complex geometries) or approximate (using a standard size for all the cells). Some of the most used exact cell decompositions are trapezoidal and Morse, both considering the critical points of the geometries, the vertex or extremes in the obstacles and environment (Choset et al., 2005). An example of approximate cell decomposition is shown in Figure 3.3, where the obstacles were approximated by square cells.

Cell decomposition methods deal with complex geometries by means of an exact decomposition, but at a very high computational cost. Also, the uses of traditional exact cell decomposition are limited to previously known static environments. Cell decomposition methods have been used previously for pursuit-evasion tasks, where the pursuers need to cover the environment to find and capture the evader (Choset et al., 2005). Approximate cell decomposition based path planning has been incorporated to the category of sample-based methods.

3.1.4 Sample-based methods

Sample-based methods consist of the computation of a map of sampled configurations of a robot in the environment (nodes), linked by collision free paths to change from one configuration to another. A solution is computed by answering a *query*, where a robot needs to move from an initial configuration to a final configuration. This family of methods has been used for manipulators, industrial automation, problems with multiple dynamic and kinematic constraints, and other planning problems different to navigation (Choset et al., 2005; LaValle, 2006; Siciliano and Khatib, 2008). Examples of sample-based methods include: probabilistic roadmaps for multiple queries, expansive-spaces trees, rapidly-exploring random trees, approximate cell decomposition in a 2-D navigation environment, and potential fields with gradient descent (where the environment is sampled in a grid).

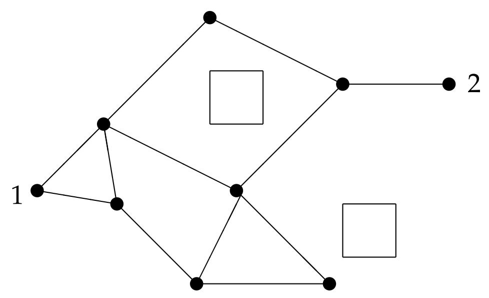


Figure 3.4: Probabilistic roadmap between the points 1 and 2 (initial and goal) in the free-space, for an environment with two square obstacles.

An example of a probabilistic roadmap is presented in Figure 3.4, where the points represent randomly sampled configurations (in this case positions in a 2-D plane), and the lines are the paths that link the configurations without colliding with obstacles. A query to go from point 1 to point 2 is the connection of paths within the roadmap.

Sample-based methods allow obtaining a path to the goal in bounded time, but are not necessarily the best option when using an exact representation of the possible configurations in the environment. As only few samples of the possible configurations are used for the planning, the computational time is reduced. Sampling offers an alternative for environments with complex geometries, and high dimensional configuration spaces in robots. The main drawback is that sample-based methods require designing different modules within the whole algorithm to ensure that any path between configurations does not collide with obstacles, and also post-processing to optimise the obtained path. Furthermore, they do not compute global optimal paths, but paths that connect an initial and a final configuration.

3.1.5 Potential functions

Potential functions can be used to represent the environment, as the path is desired to move away from obstacles and towards a goal thus repulsion and attraction functions are used correspondingly (Hwang and Ahuja, 1992a; Bell, 2005; Choset et al., 2005; LaValle, 2006; Siciliano and Khatib, 2008). Figure 3.5 represents the concepts of attraction and repulsion forces. Artificial potential fields have been widely used in path planning for both manipulator robots and for autonomous navigation in a 2-D and 3-D world. They were introduced by Khatib (1986) for robot path planning. A path is traditionally found by following the negative gradient towards the goal, although other proposals introduce search methods and using navigation functions to overcome local minima problems.

Potential functions have been applied to local partially observable environments, in a manner similar to sensor-based bug algorithms. A great portion of the research on path planning using potential functions considers a sensed, previously unknown, partially observable environment. Other path planning methods are not suitable for bug-like online implementations, where the planning is done while navigating in a partially visible environ-

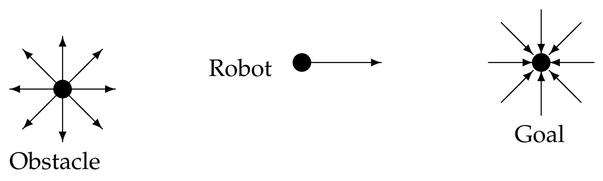


Figure 3.5: Representation of the attraction and repulsion potential forces exerted by a goal and an obstacle.

ment from sensed information. Two of the main advantages of potential functions over other path planning methods are the physical principles behind them, which make them easier to understand, and their low computational cost (Hui and Pratihar, 2008). Also, potential functions can be used in combination with other path planning and artificial intelligence techniques, like cell decomposition, sample-based planning approaches, reactive controllers, or even reinforcement learning (Megherbi and Malayia, 2012). The main drawback of using potential functions is the presence of local minima and the oscillations in closed spaces (ravine problem), mentioned in more detail in Subsection 3.3.6.

The advantages mentioned above, combined with their proven applicability to sensed dynamic environments, make potential functions a candidate method to use in a first attempt to solve the problem of navigation and escape from pursuers. Although it is desirable to use different path planning methods, combined with different search algorithms and environment representations to produce the best possible outcome, a first approach to solve the problem of navigation and escape from pursuers is done using potential functions. Section 3.3 gives a deeper insight into potential functions, including a brief historical timeline, a recollection of the main research divided into three subsections (with graphical examples), the main challenges when using potential functions (local minima), and relevant work that combines potential functions with other path planning methods.

3.1.6 Search algorithms for path planning

Search algorithms are used by path planning methods to find the connections between different possible configurations of a robot (nodes), normally from an initial configuration to a final configuration or goal (LaValle, 2006). The search can be *blind*, without an evaluating criteria, or heuristics-based where an evaluation function is used to assess if a node can generate a path towards the goal. The connections of the different related nodes are explored through a depth-first, breadth-first or any other expansion technique.

Heuristics-based search algorithms can be optimal (finding the best solution possible according to the heuristics), or suboptimal (limited to find a solution in a given time that might not be the best possible). Search algorithms imply a previous discretisation of the environment into nodes (possible configurations) using a representation according to the path planning categories mentioned before, and then a path is computed.

The most popular optimal tree search algorithms (i.e. they expand the nodes as they search) include (Bell, 2005; LaValle, 2006; Robotin et al., 2010):

- **Dijkstra algorithm**. This algorithm calculates the cost to move from a node to any other for all the nodes. The connecting cost of two nodes is the minimum moving cost possible.
- **Best-first-search algorithm**. In this algorithm, the distances between the current node and the goal are computed, choosing the next node that is closer to the goal.
- **A***. A* is a combination of the Dijkstra and the best-first-search algorithms. The minimum cost path that goes through a node is computed as the sum of the estimated minimum cost path from the start to the node, and the estimated minimum cost from the node to the goal. The search algorithm is considered *admissible* if it is able to always find the minimum cost path from the initial node to the goal. The original A* algorithm is not suited to dynamic complex environments that need constant updates in the computed paths.
- **D***. D* was inspired by A*, but targeting dynamic environments with incomplete information that can be acquired later through sensing. A graph is normally used to represent the configurations of a robot (nodes) and the costs of moving from a node to the other (connecting curves). A function, *process-state*, computes an optimal path to the goal, and another function, *modify-costs*, updates the costs in the connecting curves according to the new information from the environment. If the costs are updated, the process-state function recomputes the remaining path towards the goal.
- **Focused D***. A heuristic is incorporated into D* to update only the costs of the nodes that are more likely to generate optimal paths, thus reducing the number of expanded nodes.

The optimal search algorithms mentioned above can be bounded through combined heuristics or hierarchies to find a suboptimal solution in a given time (Luke, 2009). For example, modifications to A* like real time search LSS-LRTA*, weighted A*, IDA*.

Sample-based tree methods try to deal with continuous environments by sampling them, instead of discretising the whole set of possible configurations into nodes. Some sample-based methods include random exploration during the search for a path that connects an initial configuration to a goal. An example is rapidly-exploring random trees (Bruce and Veloso, 2002), where the path is the product of random exploration and selected configurations towards the goal within the samples.

Other popular local search algorithms (i.e. they consider a whole solution to the search formed by segments, and then adjust the segments to improve the solution) include (Russell and Norvig, 2003; Luke, 2009):

- **Hill climbing**. Starting from an initial random solution, better candidates are constructed from it, and they substitute the initial solution if they are better according to an evaluation criteria. Hill climbing is equivalent to gradient-based methods, but no mathematical functions are needed. Modifications can be added to obtain a guaranteed global optimum, like changing the amount of variation in the new candidate solutions, or random restarts in the search.

- **Random search**. This algorithm starts in a random solution, and looks for better solutions constructed from the initial solution randomly. Random search involves a larger degree of exploration than hill climbing.
- **Simulated annealing**. Also starting from a random solution, the difference between simulated annealing and hill climbing is that not always the best candidate substitutes the previous solution, but worse solutions can substitute previous with a probability. The probability decreases with time.
- **Tabu search**. In Tabu search, a record of the recent candidate solutions is kept (tabu list), and they cannot be considered as new solutions until they are allowed out of the tabu list sometime later.
- **Evolutionary and genetic algorithms**. Considering a sample of candidate solutions instead of one at a time, the candidates are evaluated and resampled to improve the final solution in a bio-inspired way.
- **Particle swarm optimisation**. This algorithm is similar to evolutionary and genetic algorithms, but the candidate solutions are modified (mutated) and not resampled.

One of the main drawbacks of the previously mentioned local search algorithms, like genetic algorithms, is the time to compute global optimal solutions. Nevertheless, limiting the time of computation allows obtaining suboptimal solutions that could be enough for the requirements of a task. Pruned tree search (A*,D*), hill climbing, simulated annealing, random search and Tabu search algorithms, have been used for path planning in robotics (including potential functions).

3.1.7 Models of the environment for path planning

Different models have been proposed for the representation of the environment in path planning. The representations, in combination with search methods, form the different classes of path planning methods mentioned in Subsections 3.1.1 to 3.1.5. A representation of the environment is required to allow the computation of an optimal or suboptimal path, using less data for the computations but still delivering an optimal solution, and calculating simpler next actions to perform (Baranauskas et al., 2008).

The models can be separated according to the main characteristics of the analysed environment: (a) models for indoor and structured environments where the ground can be considered in a plane; (b) models for natural environments with irregularities and elevations in the terrain; and (c) proposals to deal with dynamic environments or mobile elements (Siciliano and Khatib, 2008, Chapter 36). Some of the most used models are (Siciliano and Khatib, 2008):

- **Occupancy grid**. The environment is represented as a grid, and every cell has a probability of occupation by obstacles that can be recomputed according to the information available from sensor measurements.

- **Line or geometrical maps**. Considering a set of points sensed from the environment, an algorithm like split-and-merge is used to group the points into lines to represent elements in a geometrical form.
- **Topological maps**. Topological maps represent the environment from relations between the present elements, like distances between obstacles.
- **Landmarks**. The landmarks are features in the environment that can be distinguished from others. They can be represented with Gaussians in combination with extended Kalman filters, for example. The landmarks can be connected to form a path, through a relational graph.
- **Elevation grids**. A 3-D environment can be represented as a grid, if related to a plane.
- **Point sets**. A data structure that stores points of a 3-D environment is used.
- **Meshes**. Also used to represent 3-D natural environments with several irregularities.
- **Cost maps**. Points in the environment are represented in terms of the cost to move towards a specific location.
- **Semantic attributes**. Semantic attributes refer to the classification of landmarks to simplify their representation.
- **Tracking dynamic elements**. A proposed alternative to deal with dynamic environments is the tracking of movable obstacles, which is advantageous when recomputing any other model online according to the updates of information.

Grid representations have been widely used in combination with path planning techniques such as potential fields, further expanded in Section 3.3. They offer a compact useful representation for environments like houses and streets, but they can be adapted to complex elevated terrains by means of relations between cells (Siciliano and Khatib, 2008).

3.2 Stealth navigation and evasion from pursuers

Stealth navigation is an area of path planning that is related to the problem of navigation and escape, but does not offer a solution when the robot has been identified and it is being chased. Stealth navigation is based on the assumption that the opponent (pursuer, static observer or target to chase) has not found the robot in the environment. Stealth navigation offers a possibility to avoid capture if the robot reaches cover and the pursuers are unable to see it anymore and desist on their pursuit.

Covert or stealth navigation is a useful resource for tasks where detection needs to be avoided (Marzouqi and Jarvis, 2006). These tasks can be found in policing and military, like surveillance, spying, exploring hostile environments, tracking targets and crime prevention, but also in computer and video games. Different path planning methods have been used to solve the problem of stealth navigation, particularly potential functions and roadmaps.

The study of covert or stealth navigation can be grouped into three categories, according to the perspective of study:

- **General stealth**: navigating using obstacles to minimise the visibility or exposure of a robot to static observers placed in the environment. The environment is usually assumed to be known previously. For example, Teng et al. (1993) proposed using an occupancy grid morphed into a hypercube according to a visibility analysis of the elevations in a terrain, added to predicting the movements of adversaries, to plan a stealth path. Ravela et al. (1994) considered a visibility analysis of the terrain elevations in a known environment, represented with harmonic potential functions (mentioned in Subsection 3.3.2), to compute stealth paths.

  Jarvis and Marzouqi (2005) proposed finding a stealth path that considers minimising the costs according to the distance to a goal and the observability risk, in an environment approximated into a uniform grid. This is extended in Marzouqi and Jarvis (2006) for unknown and partially known environments by updating the costs of each cell as the environment is sensed. In Marzouqi and Jarvis (2005), the same authors used an exact cell decomposition in the environment.

  Birgersson et al. (2003) also worked with unknown environments, obtaining grid-based maps of terrain elevations and the *shadows* within them (where a robot is unobservable) and then computing potential functions of attraction to the shadows or the goal, and repulsion from the obstacles. Tews et al. (2004a,b) extended the work of Birgersson et al. (2003) using an occupancy grid updated with new information from sensors to represent obstacles and an observer in an unknown environment, computing respective potential functions. Tews et al. (2004b) focused on a multi-robot system where the acquired map and the traveled stealth path are shared to all robots remain covered while navigating.

- **Stealth interceptions**: navigating using obstacles to minimise the exposure to dynamic invaders, towards intercepting them at a specific location and time (Park et al., 2009), also named motion camouflage. The trajectory of the invaders is assumed to be predictable. For example, Park et al. (2009, 2010) used detection roadmaps from the environment to assess if a stealth path is available to achieve interception at a certain point and time. Other examples of motion camouflage have been mentioned in Subsection 2.2.3. This area of stealth planning deals primarily with the design of pursuers that are capable of capturing their target, whose movement is simple and can be modelled or predicted (Anderson and McOwan, 2003; Glendinning, 2004; Justh and Krishnaprasad, 2006).

- **Visibility analysis**: this area of computer science deals with the geometrical analysis of visibility behind bodies in the environment, considering obstacles, elevations in the terrain, and the embodied capabilities of the robot in terms of range and distances. For example, Woo et al. (1990) analyse different kinds of shadows formed behind bodies for different perspectives.

Some of the work considers a full knowledge of the environment (e.g. Ravela et al. (1994); Teng et al. (1993); Jarvis and Marzouqi (2005); Park et al. (2009, 2010)), including the location and shape of obstacles, and the location and range of sight of the observers (surveillance robots, fire to avoid or invaders to capture, among others), including their predictable motion if the observers are dynamic. The work on unknown dynamic environments does not consider incorporating pursuers in the planning (Birgersson et al., 2003; Tews et al., 2004b,a; Jarvis and Marzouqi, 2005), and if it does, the considerations are that the pursuers have not detected the robot initially, so stealth navigation is apparently enough as a solution (Marzouqi and Jarvis, 2006). Unfortunately those considerations are far from the case where pursuers have discovered a robot and start a pursuit, a model of the pursuers (to predict their behaviour with confidence and anticipation) is not available, and the robot has no previous knowledge of the environment.

In the context of pursuit-evasion games, path planning techniques have been used as part of the computation of strategies for evading the pursuers, as in Amin et al. (1997) and Karaman and Frazzoli (2010) oriented to aircraft. Repulsive forces exerted by the pursuers similar to potential functions (Amin et al., 1997), Voronoi diagrams (Amin et al., 1997), and incremental sampling-based algorithms (Karaman and Frazzoli, 2010) are some of the techniques used to deal with more than two pursuers. Nevertheless, most of the navigation strategies computed for pursuit-evasion games do not use path planning techniques, but optimisation theory (i.e. the computation of an optimal control law from differential equations subject to optimisation criteria) and graph search approaches (e.g. Leondes and Mons (1979); Yavin (1986); de Villiers et al. (1987); Amin et al. (1997); Fuchs et al. (2010)). Solutions to pursuit-evasion games are reviewed later in Section 6.2.

3.3 Artificial potential fields for path planning

Potential functions for path planning were introduced in the 1970s, specifically to solve problems of trajectory planning for robot manipulators with different segments and subject to kinematic constraints. The work has extended into autonomous vehicles (ground, aerial, aquatic) and even multi-robot coordination. The milestones of artificial potential fields are summarised in Table 3.1.

Potential functions have been used for the two kinds of systems that can be designed using control and path planning methods, illustrated in Figure 3.6:

- **Reactive**. In a reactive system, the perception of the environment (known *a priori* or unknown and sensing based) is mapped into potential functions that generate a vector field through the gradient operation. The gradient is then followed by the actuators in the robot. Some examples include the work of Koren and Borenstein (1991); Guldner and Utkin (1995); Tsourveloudis et al. (2001); Birgersson et al. (2003); Poty et al. (2004). Another way to implement a reactive system is through designing a control law incorporating the vector forces exerted by a potential field, as in Masoud

Table 3.1: Milestones of artificial potential fields.

YEAR	CONTRIBUTIONS
1976-1984	Khatib, Arimoto and Miyazaki, and Pavlov and Voronin introduced the possible use of potential fields for robot control (Bell, 2005).
1986	Khatib introduced a conic repulsion function and a quadratic attraction function (Khatib, 1986). Krogh and Thorpe proposed the first combination of potential fields with a search method (Bell, 2005).
1987	First topological considerations for the implementation of the potential fields by Koditschek (1987).
1989	Koren and Borenstein proposed a constant discrete attraction field that deals with uncertainty. Latombe and Barraquand combined harmonic function based potential fields with random motion search methods (Charifa and Bikdash, 2009).
1990	Connolly formulated the obtention of a potential field through a solution to Laplace's equation (Connolly et al., 1990). Search methods different to following the gradient descent are used (Bell, 2005).
1992	Rimon and Koditsheck extended the potential fields to the configuration space for path planning (Bell, 2005).
2000-2012	Dealing with multiple minima problems, narrow corridor problems, closeness of goal and obstacles problem (Charifa and Bikdash, 2009). More realistic shapes of obstacles (Bell, 2005). Other potential functions have been proposed (Charifa and Bikdash, 2009).
2002	Masoud used potential fields to solve the problem of evasion from pursuers in crowded environments (Masoud, 2002).

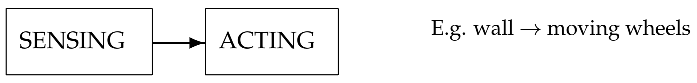
et al. (1994); Masoud and Masoud (2000); Masoud (2003b); Cherubini and Chaumette (2010); Roussos et al. (2010).

- **Deliberative**. A map of the environment using potential functions is obtained, then a search method or gradient descent is used to find an optimal path towards a goal (or temporary subgoal), for *a priori* known environments or previously unknown and sensed. The path is then tracked by a controller. Examples of deliberative controllers are the work inWarren (1989); Masoud (1996); Masoud and Masoud (2002); Masoud (2010); Pradhan et al. (2011).

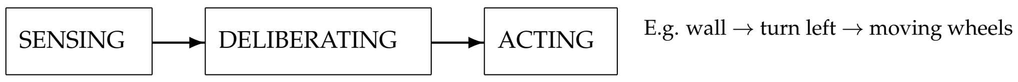
Literature on the use of potential fields for path planning can also be categorised according to the type of proposed mathematical construction, presented in Subsections 3.3.1 to 3.3.3: scalar potential functions (Subsection 3.3.1), solutions for Laplace and Poisson equations under boundary conditions (a boundary value problem) for the obstacles and the limits of the environment (Subsection 3.3.2), and obtention of a vector field directly (Subsection 3.3.3).

3.3.1 Scalar potential functions

Based on the notion of electric fields (e.g. Khatib (1986); Koren and Borenstein (1991)) or on the differences of potential in a resistor network (e.g., Tsourveloudis et al. (2001)), a scalar field can be computed from the superposition of potential functions that consider the distan,ce between the important points in the environment (obstacles and goal) and a robot. Obstacles are assumed to have a positive charge, therefore exerting a repulsion force in all directions, and the goal can be assumed to have a negative charge, exerting an attraction force towards its location (Koren and Borenstein, 1991).



(a) Reactive control system



(b) Deliberative control system

Figure 3.6: Structures of a reactive and a deliberative control systems.

Different potential functions have been proposed to represent the attraction of the goal, as well as the repulsion of the obstacles, the most used presented in the following paragraphs. The principle of superposition can be applied to combine the effect of all the forces in a field,

$$\phi(x, y) = E_1 + \cdots + E_m - E_g \tag{3.1}$$

where $\phi(x, y)$ is the scalar field for all points (x, y) in a 2-D space (that can be extended to more dimensions, $x_1, \ldots, x_l$). $E_1, \cdots, E_m$ are the potential functions of each obstacle in the environment, and E_g is the potential function of the goal. The vector field of Equation 3.1 can be obtained from the gradient of the scalar field defined by potential functions (Sowerby, 1974). A path can be computed following the gradient descent towards the goal for continuous tasks, or from other search methods if the representation of the environment is discrete (like steepest descent or the ones mentioned in Subsection 3.1.6).

Attraction functions for the goal

Some of the most popular attraction potential functions are:

- **Smoother quadratic well** (Koren and Borenstein, 1991; Ge and Cui, 2000; Vadakkepat et al., 2001; Velagic et al., 2006): a conic function has the form

$$E_g = K_g \cdot (R_g)^{1/2} \tag{3.2}$$

  where $R_g = (x - x_g)^2 + (y - y_g)^2$ and K_g representing the force of the effect of the goal in the potential field in a 2-D plane. This function has a similar shape to the electric field function.

- **Quadratic goal** (Khatib, 1986; Al-Sultan and Aliyu, 1996; Yun and Tan, 1997; Haddad et al., 1998; Ge and Cui, 2000; Huh et al., 2002; Wang et al., 2002; Masehian and Amin-Naseri, 2004; Zhu et al., 2006; Mora and Tornero, 2008; Takahashi et al., 2010; Tan et al., 2010): a quadratic representation can be used instead of Equation 3.2 for the goal,

$$E_g = K_g R_g \tag{3.3}$$

where $R_g = (x - x_g)^2 + (y - y_g)^2$ as presented in Equation 3.2 for a 2-D plane. A quadratic function has better stabilising properties when finding a solution for the path planning problem, as it helps to prevent some local minima.

- **Others**: superquadric functions (Khosla and Volpe, 1988), Gaussian functions (McIsaac et al., 2003), harmonic functions (Guldner and Utkin, 1995).

Repulsion functions for the obstacles

Some of the most popular repulsion potential functions are:

- **Quadratic well** (Koren and Borenstein, 1991; Vadakkepat et al., 2001; Velagic et al., 2006): for n_o obstacles represented as points in space, the electric field for every one of them can be approximated as

$$E_m = \frac{K_m}{R_m + c} \qquad \forall m \in \{1, \ldots, n_o\} \tag{3.4}$$

  where $R_m = (x - x_m)^2 + (y - y_m)^2$, K_m representing the force of the effect of the mth obstacle in the potential field in a 2-D plane, and c a constant that avoids a division by zero when the distance is close to zero. If the robot collides with an obstacle, i.e. $(x, y) = (x_m, y_m)$, the total field approximates a large bounded value. The values of the constants $K_1, \ldots, K_{n_o}$ affect the magnitude of the repulsion force and therefore they need to be selected according to the requirements of the task.

- **Limited quadratic function** (Wang et al., 2002; Ding et al., 2005; Zhu et al., 2006, 2009; Lee et al., 2010; Takahashi et al., 2010; Tan et al., 2010; Pradhan et al., 2011): a modification of the conic function was proposed to limit the influence region of the obstacles in the environment,

$$E_m = \begin{cases} K_m \left(\frac{1}{f(x,y)} - \frac{1}{f(x_0,y_0)} \right)^2 R_g & \text{if } f(x,y) \leqslant f(x_0,y_0) \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

  where $f(x, y)$ is a function of the distance, or the velocity and the distance, and $f(x_0, y_0)$ is the limit of the influence of the obstacle in a 2-D plane. The term R_g, same as in Equations 3.2 and 3.3, deals with situations where a local minimum is present due to an obstacle and the goal being equidistant to the robot, but some authors do not use it resulting in the so-called FIRAS (Force Inducing an Artificial Repulsion from the Surface, in French) function (Khatib, 1986; Janabi-Sharifi and Vinke, 1993; Yun and Tan, 1997; Ge and Cui, 2000; Mora and Tornero, 2008).

- **Solid obstacles** (Al-Sultan and Aliyu, 1996; Hwang and Ahuja, 1992b): obstacles can be represented by solid shapes,

$$S_m = x, y : g_m(x, y) \leq 0; g_m(x, y) \in C^2[0, \infty), (x, y) \in \mathbb{R}^2 \tag{3.6}$$

  where $g_m(x, y)$ is the shape (e.g. circle) of the contour of the mth obstacle in a two-dimensional plane, and C^2 is the class of the function $g_m(x, y)$ (differentiable twice and the derivatives are continuous). Al-Sultan and Aliyu (1996) used a cylindrical field for each obstacle, inside the region described by S_m, of the form

$$E_m = K_m \max\left(0, -g_m(x, y)\right)^r \tag{3.7}$$

where K_m is a constant of large value that defines the strength of the field, and $r \geq 2$, for $m = 1, \ldots, n_o$ obstacles.

- **Others**: superquadric functions (Khosla and Volpe, 1988), fractional functions (Poty et al., 2004), elliptical (Pradhan et al., 2011), Gaussian (McIsaac et al., 2003), dipolar (Roussos et al., 2010), exponential (Weir and Bott, 2010), sigmoid (Ren et al., 2007), circular (Pathak and Agrawal, 2004).

Many authors have used the proposals formulated by Khatib (1986) in their research. Conic repulsion and quadratic attraction potential functions are the best alternatives when combining approaches against local minima that are easy to implement computationally. The formulations in terms of distance can be adapted to sampled points in the configuration space, or even in a grid by using the centroids of the cells as reference points for the functions. This allows dealing with navigation problems where the geometry of the environment is too complex geometrically, or even unknown like in analysis of the environment mapping that are based on information from sensors.

3.3.2 Boundary value problem solutions to the Laplace equation

In order to avoid local minima, one of the main challenges when designing potential functions, some researchers have proposed finding functions that respect the conditions of the Laplace equation as they have a unique minimum in the location of the goal (Connolly et al., 1990; Kim and Khosla, 1992; Masoud et al., 1994; Masoud and Masoud, 2000; Prestes et al., 2001; Masoud and Masoud, 2002; Alvarez et al., 2003; Masoud, 2003a; Trevisan et al., 2006; Pimienta et al., 2006; Shi et al., 2007; Silveira et al., 2010; Masoud, 2010). Considering the Laplace equation (a harmonic function) of a field of l dimensions, $\phi(x_1, \ldots, x_l)$,

$$\sum_{l=1}^{L} \frac{\partial^2 \phi}{\partial x_l^2} = 0 \tag{3.8}$$

the resulting field does not have problematic local minimum points. If a function satisfies Equation 3.8 in a region, any critical point within the region is a saddle point. If the robot reaches the saddle point far from the goal, it can find a path out of it.

The potential field is obtained by formulating a boundary value problem (BVP) for a differential equation like the Laplace. In a BVP, boundary conditions are proposed for the obstacles, the limits of the environment and the goal. An example is the use of Dirichlet conditions, where traditionally the following values are used: $\phi(x, y) = 1, \forall x, y \in m, m = 1, \ldots, n_o$ for the locations or boundaries of the n_o obstacles and the limits of the environment, and $\phi(x, y) = 0$ for the goal regions (Connolly, 1994). Other conditions that can be used in the BVP are Neumann (Zelek, 1995) or anisotropic (Masoud, 2002).

A path can be obtained by following the resulting vector field (e.g. in Prestes et al. (2001); Mbede et al. (2002); Masoud (2002, 2003b)). Vector solutions to the boundary value problem can also be used to obtain control laws as in Masoud et al. (1994); Masoud and Masoud (2000, 2002); Masoud (2007a).

The Laplace equation (Equation 3.8) can be approximated by a Taylor expansion. An approximated scalar potential field can be represented by a grid, where every inner cell x_l has a potential value defined as an average of the surrounding cells (stencil). Calculating the Taylor series expansions of the gradient about the point x_1 gives

$$\phi(x_1+\delta,\ldots,x_l) = \phi(x1,\ldots,x_l) + \delta\frac{\partial\phi}{\partial x_1} + \frac{1}{2}\delta^2\frac{\partial^2\phi}{\partial x_1^2} + \frac{1}{6}\delta^3\frac{\partial^3\phi}{\partial x_1^3} + \mathcal{O}(\delta^4) \tag{3.9}$$

$$\phi(x_1-\delta,\ldots,x_l) = \phi(x1,\ldots,x_l) - \delta\frac{\partial\phi}{\partial x_1} + \frac{1}{2}\delta^2\frac{\partial^2\phi}{\partial x_1^2} - \frac{1}{6}\delta^3\frac{\partial^3\phi}{\partial x_1^3} + \mathcal{O}(\delta^4) \tag{3.10}$$

for small δ.

The two expansions are added,

$$\phi(x_1+\delta,\ldots,x_l) + \phi(x_1-\delta,\ldots,x_l) = 2\phi(x1,\ldots,x_l) + \delta^2\frac{\partial^2\phi}{\partial x_1^2} + \mathcal{O}(\delta^4) \tag{3.11}$$

and after some algebraic manipulation, considering $\mathcal{O}(\delta^4)\approx 0$

$$\frac{\partial^2\phi}{\partial x_1^2} = \frac{\phi(x_1+\delta,\ldots,x_l)+\phi(x_1-\delta,\ldots,x_l)-2\phi(x_1,\ldots,x_l)}{\delta^2} \tag{3.12}$$

If the same procedure with the Taylor series expansions is applied for every variable $x_1,\ldots,x_l$, an approximation of Equation 3.8 can be obtained

$$\sum_{l=1}^{L}\frac{\partial^2\phi}{\partial x_l^2} = \frac{1}{\delta^2}\Bigg[\phi(x_1+\delta,\ldots,x_l)+\ldots+\phi(x_1,\ldots,x_l+\delta)+\phi(x_1-\delta,\ldots,x_l)+\ldots \\ +\phi(x_1,\ldots,x_l-\delta)-2l\phi(x_1,\ldots,x_l)\Bigg] = 0 \tag{3.13}$$

and after more algebraic manipulation gives

$$\phi(x_1,\ldots,x_l) = \frac{1}{2l\delta^2}\Bigg[\phi(x_1+\delta,\ldots,x_l)+\ldots+\phi(x_1,\ldots,x_l+\delta)+\phi(x_1,\ldots,x_l+\delta)+\ldots \\ +\phi(x_1,\ldots,x_l-\delta)\Bigg] \tag{3.14}$$

where the value of the field $\phi(x,y)$ at (x,y) is the average of the values of the points in the immediate vicinity. For a 2-D environment, (x,y), and with a distance of 1 from the point or cell to the next (i.e. $\delta=1$), the potential in a point is reduced to

$$\phi(x,y) = \frac{1}{4}[\phi(x+1,y)+\phi(x,y+1)+\phi(x-1,y)+\phi(x,y-1)] \tag{3.15}$$

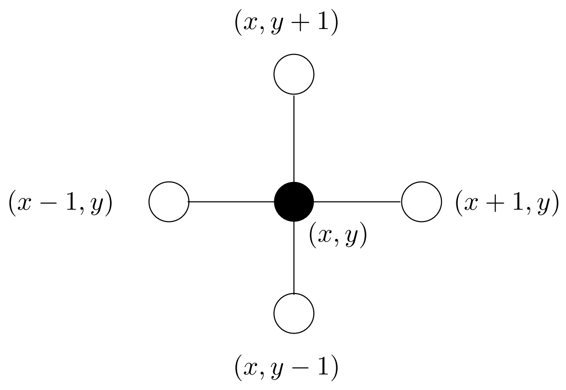


Figure 3.7: Five points stencil from Equation 3.15.

also known as a five points stencil, represented in Figure 3.7.

The potential field stencils are calculated by the formulation of a BVP solved by numerical methods, particularly relaxation. Considering a square grid of $n \times n$, rows and columns respectively, where every internal cell has a value of $z_{a,b}$, the four points stencil from the Taylor series expansion of the Laplace equation is

$$z_{a,b} = \frac{1}{4}[z_{a-1,b} + z_{a,b-1} + z_{a,b+1} + z_{a+1,b}] \tag{3.16}$$

The row of the cell is denoted by the subscript a and the column by b. Each equation can be written as a linear combination of the value of every other cell in the grid, but with coefficients zero in the terms that are not contained in the stencil, for example

$$\begin{aligned} z_{2,2} = \frac{1}{4}[0 \cdot z_{1,1} + 1 \cdot z_{1,2} + 0 \cdot z_{1,3} + \ldots + 0 \cdot z_{1,n} + 1 \cdot z_{2,1} + 0 \cdot z_{2,2} + 1 \cdot z_{2,3} + 0 \cdot z_{2,4} + \ldots \\ + 0 \cdot z_{3,1} + 1 \cdot z_{3,2} + 0 \cdot z_{3,3} + \ldots + 0 \cdot z_{n,n}] \end{aligned} \tag{3.17}$$

The set of equations that describes the value of each internal cell in the grid, expressed in the form of the previous equation, can be grouped in vectors and matrixes,

$$\begin{bmatrix} z_{1,1} \\ \vdots \\ z_{2,2} \\ \vdots \\ z_{n,n} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \frac{1}{4} & 0 & \ldots & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & \ldots & 0 & \frac{1}{4} & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 1 \end{bmatrix} \begin{bmatrix} z_{1,1} \\ \vdots \\ z_{2,2} \\ \vdots \\ z_{n,n} \end{bmatrix} \tag{3.18}$$

This set of linear equations can be also written as

$$\vec{z} = \frac{1}{4}\mathbf{C}\vec{z} \tag{3.19}$$

where C is a matrix that indicates the terms that form the stencil of each internal cell in the grid by a magnitude of 1 or 0 if not. To obtain the values of the approximated grid that now represents the 2-D plane, a boundary value problem is proposed where values are assigned to $\vec{z}$, including the edges and critical points (the obstacles and the goal). A relaxation algorithm computes the updates for $\vec{z}_{new}$ from $\vec{z}_{old}$ by iterations, minimising the total error until convergence to a threshold ϵ_{th}.

Relaxation algorithms include Jacobi, Gauss-Seidel, red-black ordering, and successive over-relaxation (SOR). Two relaxation algorithms have been used frequently in the computation of the stencils for the BVP: the Gauss-Seidel algorithm (Connolly et al., 1990; Zelek, 1995; Trevisan et al., 2006; Vaščák, 2007), and SOR (Alvarez et al., 2003; Shi et al., 2007; Silveira et al., 2010), the latter is presented below along with the Jacobi algorithm in Algorithms 1 and 2. Pimienta et al. (2006) solved the BVP using the finite element method. Other alternatives to the Laplace equation for the BVP include the Poisson equation (Masoud, 2003a,b), the Navier-Stokes equation for viscous incompressible fluids (Louste and Liegeois, 2000), the diffusion equation (Charifa and Bikdash, 2009) and the biharmonic equation (Charifa and Bikdash, 2009).

Algorithm 1 Successive over-relaxation (SOR) algorithm for the BVP

```
1: Initialise $\vec{z}_{old}$ with a guess, like $z_{a,b} = 0.5, \forall a, b$. $\lambda_{SOR} = \frac{4}{2+\sqrt{4-(\cos\frac{pi}{a}+\cos\frac{pi}{b})^2}}$
2: Apply boundary conditions to $\vec{z}_{old}$: $z_{a,b} = 1$ for all cells that are obstacles, $z_{a,b} = 0$ for
   the goal, and $z_{a,b} = 0.5$ for all edges of the grid
3: Compute $\vec{z}_{new} = \vec{z}_{old} + \lambda_{SOR}\mathbf{C}\vec{z}_{old}$
4: Compute $error = \frac{1}{n^2}\sum_{a,b}\left(z^{new}_{a,b} - z^{old}_{a,b}\right)$
5: if $error > \epsilon_{th}$ then
6:     $\vec{z}_{old} \leftarrow \vec{z}_{new}$
7:     Repeat from 2
8: end if
```

Algorithm 2 Jacobi iteration algorithm for the BVP

```
1: Initialise $\vec{z}_{old}$ with a guess, like $z_{a,b} = 0.5, \forall a, b$
2: Apply boundary conditions to $\vec{z}_{old}$: $z_{a,b} = 1$ for all cells that are obstacles, $z_{a,b} = 0$ for
   the goal, and $z_{a,b} = 0.5$ for all edges of the grid
3: Compute $\vec{z}_{new} = \mathbf{C}\vec{z}_{old}$
4: Compute $error = \frac{1}{n^2}\sum_{a,b}\left(z^{new}_{a,b} - z^{old}_{a,b}\right)$
5: if $error > \epsilon_{th}$ then
6:     $\vec{z}_{old} \leftarrow \vec{z}_{new}$
7:     Repeat from 2
8: end if
```

3.3.3 Vector fields

Designing vector fields in a direct form allows the incorporation of rotational and more complex forces to guide the motion of a robot in a reactive controller, or to compute a path for a deliberative system by following the vectors in the field. Examples include incorporating rotational tangent forces to avoid collisions with obstacles or other robots in a multi-agent system (Slack, 1993; Masoud, 1996; Haddad et al., 1998; Masoud, 2007a; Ohnishi and

Imiya, 2008), incorporating vortex fields for the obstacles (McInnes, 2003; Cherubini and Chaumette, 2010), using only attractive fields exerted by the obstacles towards the goal (de la Paz López et al., 2009), and using multiple vectors to determine the attractive and repulsive forces (Jing and Wang, 2004; Cen et al., 2007). Other proposals constructed vector fields from other mathematical models, like the density of flow (Wong and Spetsakis, 2000).

Considering the same idea of incorporating additional features in the potential functions, some researchers have proposed designing control laws based on vector potential fields with rotational properties that allow correcting the path as the environment changes, when kinematic and dynamic constraints are needed, as in the case of manipulators (Khatib, 1986; Rimon and Koditschek, 1992; Masoud and Masoud, 2000). Vector potential fields can be obtained from a BVP through the process in Subsection 3.3.2. Other approaches presented the design of a control law considering a realistic dampening effect due to the inertia of a robot, from a vector field (Masoud, 2007b, 2009). Huang et al. (2006) proposed a steering control (angular acceleration) to avoid obstacles, considering attraction and repulsion as functions of the angles and distances between the robot and obstacles or the goal.

The following subsections mention some proposals to deal with dynamic environments within the reviewed literature on potential functions, the use of discretisations as a representation of the environment, and finally the main challenges when implementing potential functions for path planning: local minima, plateau regions, saddle points and ravines.

3.3.4 Planning for dynamic environments and information from sensors

Commonly used solutions for partially known environments (also referred to as dynamic environments in the literature), with dynamic obstacles, previously unknown environments, and sensed information involving potential fields are: reactive approaches (bug-like), and gathering more information from the environment with sensors and constantly re-planning the path until the objective has been fulfilled (Bruce and Veloso, 2002; Giesbrecht, 2004).

Reactive control systems have been used recurrently for navigation in a sensing based setting (Koren and Borenstein, 1991; Vadakkepat et al., 2001; Mbede et al., 2002; McIsaac et al., 2003), often combined with an occupancy grid representation, where obstacles and free space are mapped into cells. Combined with potential functions, the actions taken by the robot are dictated by following the forces of the potential field. If scalar functions are used to represent the environment, the gradient is computed. The navigation actions are implemented using a low level controller like neuro-fuzzy based (Mbede et al., 2002) or PID (Birgersson et al., 2003). Using reactive potential functions based control systems has the main disadvantage of a lack of look-ahead, which can lead to non-optimal paths and oscillations in the navigation (Bruce and Veloso, 2002). An alternative is using an online planning approach with updates to the path according to the incoming information from the environment.

Also, direct control based approaches, i.e. control laws that incorporate potential forces, deal with sensed previously unknown environments. Deliberative systems derived from scalar functions, solving the BVP and direct vector fields have been designed more frequently in recent times to deal with dynamic obstacles and information from environments from sensors. Examples of control laws that incorporate forces are mentioned in Weir and Bott (2010); Cherubini and Chaumette (2010); Tan et al. (2010); Takahashi et al. (2010); Pradhan et al. (2011).

3.3.5 Discrete potential fields

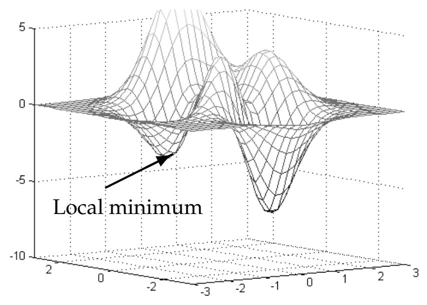
Grid representations simplify the elements in the environment, as obstacles might have a complex geometry. Occupancy grids have been used to represent the elements of the environment, the obstacles and goal, for example in Koren and Borenstein (1991); Borenstein and Koren (1991); Tsourveloudis et al. (2001); Wang et al. (2002); Birgersson et al. (2003); Shem et al. (2008); Cherubini and Chaumette (2010); Zhang and Gao (2011). Other cell representations include Haddad et al. (1998); Prestes et al. (2001). A potential field is calculated for cells in a grid, generally as scalar values, and then a path is found using discrete gradient descent (steepest descent) or discrete search methods like the ones mentioned in Subsection 3.1.6 (Bell, 2005; Choset et al., 2005). More grid discretisations are present in the numerical solutions to the BVP for Laplace's equation as in Subsection 3.3.2 (Connolly et al., 1990; Alvarez et al., 2003; Pimienta et al., 2006; Vaščák, 2007; Silveira et al., 2010).

3.3.6 Challenges implementing potential functions

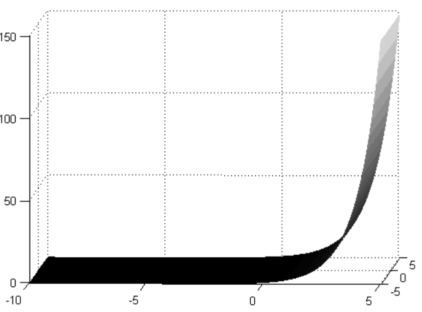
The main challenges when implementing artificial functions, and addressed by some researchers, are the following (Al-Sultan and Aliyu, 1996; Ge and Cui, 2000; Zhu et al., 2006):

- The functions should not contain local minima.
- The functions should mimic obstacle surfaces, to maximise the free-space for the path planning, but still being able to navigate through narrow passages without oscillations.
- The influence of an obstacle should be limited to its vicinity, so that goals are reachable even with nearby obstacles.
- A scalar potential field should be described as a continuously differentiable function, so the gradient can be computed.

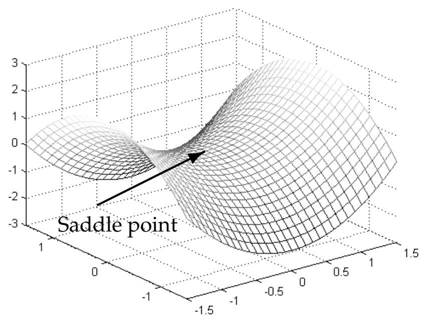
Local minima are points in the surface of the potential field, where the gradient of all the surrounding points descends towards them. Other problematic topologies that can be considered as local minima are the plateau, a plane region surrounded by points with descending gradient towards it, and saddle points, where the gradient is undefined but it descends in some points around it so an exit is possible (Bell, 2005). Visual examples of the mentioned problematic topologies are shown in Figure 3.8.



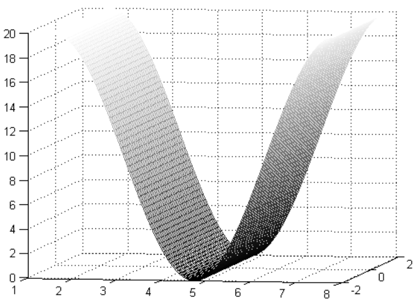
(a) Local minimum



(b) Plateau region



(c) Saddle point



(d) Ravine

Figure 3.8: Common problems when using potential functions: a local minimum on the left of the graph, a plateau region, a saddle point at $(0, 0)$ and a ravine problem.

Two main groups of solutions to the problem of local minima in potential functions have been proposed: incorporating search methods or random navigation to find a path that leads out of local minima, or finding a navigation function with one local minimum only (the goal). Within the first group are random walks (Barraquand et al., 1992; Chang, 1996; Caselli et al., 2002), following walls (Yun and Tan, 1997; Zhu et al., 2009), depth-first (Al-Sultan and Aliyu, 1996), temporal local minima (Bell, 2005), simulated annealing (Janabi-Sharifi and Vinke, 1993; Zhu et al., 2006), and the wave front planner (Giesbrecht, 2004; Choset et al., 2005).

The use of the Laplace equation and a BVP to compute an approximated vector or scalar field (Subsection 3.3.2) has been proposed to solve the problems of local minima (Prestes et al., 2001; Mbede et al., 2002; Masoud, 2002, 2003b; Alvarez et al., 2003; Vaščák, 2007). Other approaches propose modifications in the design of the potential functions to achieve only one local minimum, or at least reducing the overall number of critical points (navigation functions), include: using elliptical functions, cutting off obstacles from the surface, a sphere approach to model obstacles and the environment (Koditschek, 1987), star space that models any polygonal obstacle into spheres (Rimon and Koditschek, 1992; Choset et al., 2005), Gaussian functions, harmonic functions (Connolly et al., 1990; Masoud, 2003a), and quadratic functions (Khatib, 1986; Al-Sultan and Aliyu, 1996; Masehian and Amin-Naseri, 2004). Modifying the design of the potential field can reduce the presence of local minima (Bell, 2005).

Other proposals outside of the two main groups include converting local minima into virtual obstacles with a repulsion force (Park and Lee, 2003), the use of subgoals (Zou and Zhu, 2003; Bell, 2005; Weir et al., 2006), or adding an extra force in the field to avoid falling in the local minima (Ding et al., 2005; Lee et al., 2010).

The ravine problem or the presence of narrow passages between obstacles, illustrated in Figure 3.8, could cause an unbounded bouncing navigation movement between the two obstacle walls. This problem has been solved by using discrete approaches, mentioned in the previous subsection (Bell, 2005).

3.3.7 Combining potential functions with other path planning methods

Potential functions for path planning have not been frequently combined with other path planning methods (mentioned previously in Section 3.1). The few examples in the literature include using Voronoi diagrams (Masehian and Amin-Naseri, 2004) and graph search (Warren, 1989). These combinations have been proposed to avoid the problem of local minima inherent in the potential functions.

3.4 Discussion

Real life navigation problems require dealing with previously unknown, sensed environments, with incomplete information (partially observable) and dynamic components. When considering a path planning approach for autonomous navigation in dynamic environments, a constant update of the path is needed to succeed in the task of navigation. For these reasons, the following path planning mechanisms have been considered for the designs in this thesis:

- Applying a cell decomposition to the sensed partially visible environment allows a representation of the environment that is not precise nor exact, but still gives enough information on the locations of the obstacles and pursuers at an instant of time, when the sensing is taking place. Furthermore, a cell decomposition can be interpreted as a sample-based approach, where sensed points in the environment are used to obtain a representation of the environment and subsequently an appropriate path.

- Using potential functions based methods to guide the decision making towards the planning by assigning attraction and repulsion forces to the elements in the environment. A grid based representation can be combined with potential functions methods, as explained throughout the chapter.

Many other specific methods and algorithms, like Voronoi diagram based roadmaps, can be adapted to solve the problem of navigation and escape from pursuers in a dynamic environment. Nevertheless, potential functions are easier to understand and implement in terms of geometrical analysis. Furthermore, many of the approaches on escape from pursuers and stealth navigation have been developed from potential field methods (Masoud, 2002, 2003a,b), thus comparisons on the evaluation on the approaches for the solution of the problem of navigation and escape from pursuers can be drawn.

Although the problem of escape from pursuers has been studied for aircraft in a 3-D world (Amin et al., 1997) and as part of pursuit-evasion games (computing strategies for both pursuit and evasion simultaneously, reviewed in Chapter 6), the design of intelligent escape from pursuit of ground vehicles in a 2-D plane has not been studied with the same interest. Some of the only research efforts include the work in Masoud (2002, 2003a,b) and Karaman and Frazzoli (2010), and the study of covert navigation in Marzouqi and Jarvis (2006) from a path planning point of view. Covert navigation does not solve the problem of escape from pursuit when the pursuers have discovered a robot and the robot is not in hide. Why is there a lack of interest in researching the design of intelligent escape from pursuit? Adding intelligent capabilities to a fully autonomous robot to make it react during endangering situations allows implementing a form of self-preservation beyond avoiding obstacles (Marzouqi and Jarvis, 2006). Applications in surveillance, search and rescue, reconnaissance, guidance and interaction with humans and animals suggest that there is an area of opportunity, for the implementation of evasion from pursuit as self-preservation behaviours in the navigation.

The problem of evasion from pursuers has been studied by Masoud (2002, 2003a,b) using a potential functions approach for the navigation and escape, but it is based on cluttered environments. The solutions use a boundary value problem for the Laplace or Poisson equations, and produce vector or scalar potential fields that can be used to compute a path following the gradient descent or by incorporating them into a control law.

The design of scalar fields from potential functions or from BVPs has many drawbacks in terms of the presence of different kinds of local minima, the selection of parameters regarding the strength of the forces of attraction and repulsion and their limits of action, and dealing with complex geometries. Vector fields allow the incorporation of rotational and vortex forces, which could help to achieve the desired actions of fleeing and proteanism when navigating from the planning stage.

Finally, a high level control system is more desired than one that integrates a high level and low level, like the ones presented in Khatib (1986) or Masoud and Masoud (2000), as it can be adapted to any mobile robot without the need to know the equations that model the exact platform (low level). The adaptation of a high level into a low level control system that governs the physical components that allow the navigation following a designed path becomes a shorter step if some of the constraints (like the size of the robot) are considered during the planning of the path. The proposals contained in this thesis contemplate considering kinematic constraints, but the premise is to design a high level control system that can be used with any low level controller to achieve navigation and escape from pursuers independently from the used mobile robot.

3.5 Concluding remarks

The chapter presented a brief literature review on the area of path planning, mentioning the main groups of methods and their general properties: sensor-based bug algorithms, roadmaps, cell decomposition, sampling-based methods, and potential functions. The main discrete search algorithms used within the path planning methods were mentioned, along with the main proposed models of the world for different kinds of environments. This is followed by a summary on stealth navigation research, and other previous work on evasion, from the path planning perspective.

The literature review illustrates the main differences between the path planning techniques, being potential functions the one chosen for the design in this thesis in combination with a discretisation in space and time of the sensing of the environment, inherent of different kinds of sensors in robots. Potential functions have been used for stealth navigation and evasion of aircraft, due to the advantages of this path planning technique over others more computationally expensive. Nevertheless, different kinds of potential functions have not been compared in an evasion task. Furthermore, the use of different path planning techniques to tackle the problem of autonomous navigation and escape from pursuit is limited to some examples.

Subsequently, the chapter presented a comprehensive literature review on potential functions to illustrate the state of the art of the topic. The literature was analysed from different characteristic features:

- A reactive or deliberative control approach for the navigation of a robot.
- The design of the potential functions or fields:

  - Scalar potential functions, where different proposed functions for obstacles and the navigation goal are mentioned.
  - Solutions to boundary value problems for the Laplace equation, used to compute vector fields, or scalar fields through a numeric approximation.
  - Vector fields from design, incorporating rotations and vortices, or for computing control laws.
- Approaches that deal with dynamic sensed environments, where the information is incomplete.
- Approaches that incorporate discretisation in the obtention of potential fields.
- Approaches that deal with the main challenge when implementing potential functions: the presence of local minima, by modifying the design of navigation functions or by incorporating different search methods.

The different potential functions offer solutions to the local minima problems. Some of them have been chosen to be adapted to the designs presented in the thesis, due to the simplicity of their calculation for the fast computation of solutions in a robotic platform.

Finally, a discussion of the presented literature review takes place, highlighting: (a) the reasons behind the selection of a potential function based approach with a cell decomposition sampling for the designs presented in the thesis, and (b) the characteristics, advantages and disadvantages of the potential function methods reviewed towards the solution of the problem of navigation and escape from pursuers.

The next chapter presents a first approach to the solution of the problem of navigation and escape from pursuit. The problem to be solved with the designs is discussed first. Then, proposals for the analysis of the environment using different kinds of potential functions are described, discussed and exemplified. The path planning concepts regarding potential functions provided in this chapter serve as the basis for the designs presented in Chapter 4.

Chapter 4

Bio-inspired analysis of the environment and path planning

This chapter presents the design of solutions to the problem of navigation and escape based on a path planning approach using potential functions. The first part introduces the reasoning behind the selection of the combined specific approaches, including the definition of the problem, the use of potential functions in a discrete setting, and the bio-inspired features.

Section 4.1 presents the definition of the problem solved in this thesis, including the main elements and particularities that state the differences between the problem of escape from pursuit and other similar problems like stealth navigation. Section 4.2 discusses the main decisions regarding the characteristics of the control approach, the methods used in the designs to solve the problem of navigation and escape from pursuers, and other basic assumptions for pursuers, the environment and the validation of the results. The section also includes the methodology followed for the designs.

Section 4.3 details the biologically inspired features added in the designs, gathered from the analysis of related literature in Chapter 2. The features have been incorporated in two different forms: *a priori* and *a posteriori*. The section also presents the general structure used for the designs of solutions to the problem of navigation and escape from pursuers, from the general methodology proposed in Section 4.2 and including the biologically inspired features.

The second part of the chapter presents the detailed designs, divided in two main segments: the analysis of the environment with incorporated bio-inspired features, and then the computation of a path. The analysis of the environment is contained in Section 4.4, computing a representation that makes possible finding a navigation path that avoids obstacles and capture by pursuers simultaneously from a proposed methodology described in the corresponding subsections. Subsection 4.4.1 initialises the analysis of the environment with the approximation of the sensed elements into a grid. Subsection 4.4.2 introduces the computation of hideaways from the shadows cast by obstacles out of the sight from pursuers, representing bio-inspired refuges. Subsection 4.4.3 presents the computation of a function of the distances between the different elements in the sensed environment, to find the best locations away from pursuers. Subsection 4.4.4 introduces the computation of a local navigation goal as a function of the distances to the pursuers and the possible bio-inspired hideaways (refuges). Subsection 4.4.5 presents a proposal to compute bio-inspired subgoals for *a priori* proteanism. Subsection 4.4.6 proposes how to update the environment as the robot moves and the pursuers respond. Subsection 4.4.7 discusses the use of a finer resolution grid.

Subsequently, the adaptation of different potential function approaches to the problem of navigation and escape are presented in Section 4.5: discrete and popular potential functions in Subsection 4.5.1, approximated solutions to a boundary value problem in Subsection 4.5.2, and wave-front potential functions in Subsection 4.5.3. The subsections include examples of resulting functions in randomly generated environments. Subsection 4.5.4 presents a comparison of the proposed modified potential functions.

Next, the methods for the computation of a path are presented. Section 4.6 describes the approach to compute the path from the potential functions with steepest descent, including the algorithm and visual examples. The section also describes the implementation of bio-inspired proteanism *a posteriori* in a previously computed path.

Finally, Section 4.7 includes the discussion of the designs presented in the chapter, outlining their advantages, disadvantages and implementation challenges. The conclusions of the chapter are presented in Section 4.8.

4.1 The navigation and escape problem

The problem of navigation and escape from pursuers has been defined, for the purposes of this thesis, as consisting of a robot avoiding capture whilst also avoiding collision with obstacles. In this thesis the problem is solved for a robot navigating in a 2-D environment, but the proposals can be extended to 3-D for aerial vehicles. Capture means that the robot is surrounded by pursuers, or when the distance from any pursuer to the robot is minimum. A solution to the problem is a set of navigation actions that *(a)* avoid collision with obstacles, and *(b)* increase the distance between the pursuers and the robot or achieve getting out of the vision field of the pursuers.

The elements included in the problem are:

1. **Environment**: a locally observed portion of terrain, $E(t)$, with n_o obstacles and n_p pursuers. The environment is stochastic regarding the strategies adopted by the pursuers, as only some assumptions can be made about their behaviour from observation (sensing), but without the certainty that they are true. The environment can extend beyond the currently observed part.
2. **Obstacles**: assumed to be static elements in the environment that the robot needs to avoid to prevent collisions.

3. **Robot (the prey)**: a mobile robot with limited sensing of its surroundings. The robot needs to navigate according to what it senses if its memory is limited, and if no mapping capabilities are included in parallel to the navigation and evasion from pursuers. The robot navigates in a 2-D plane with maximum velocity, v_{max}, and acceleration, a_{max}.

4. **Pursuers**: other robots, animals, people or vehicles chasing the robot, i.e. navigating towards it, or looking for it in the environment. They move with maximum velocity, v_p^{max}.

Distances between the robot and any other object can be computed when the object is observed by the robot. The robot decides what to do next in response to its analysis of the environment. The environment, particularly the pursuers, responds to the actions of the robot (Ring, 1994). Thus, the environment is considered dynamic.

The problem of navigation and escape from pursuers is particular and different to other problems in the following aspects:

- It is not a problem that involves only avoiding collisions with obstacles or other robots, but the pursuers try to chase and possibly destroy the robot.

- It is not a stealth problem as the robot is in direct sight of some of the pursuers, and they can communicate the current position of the robot to all the pursuers in the environment. Furthermore, the robot does not know the environment in its totality, and it may be able to observe only parts of it, so it cannot plan a long distance stealth path, but only respond to the current situation of the observed environment.

- If the pursuers were static observers, the problem of being detected and running away could be solved simultaneously by navigating towards a hideaway and then continuing in a stealth mode navigating behind obstacles. But pursuers can be other intelligent enemy robots with mapping, planning, predicting and destroying capabilities.

- Fleeing away from the pursuers in a straight line or any other predictable linear movement is disadvantageous against intelligent pursuers that can do stealth pursuit, motion camouflage, or any sort of interception.

Figure 4.1 shows a visual representation of three different problems: the problem of avoiding obstacles only; the problem of stealth navigation with static observers; and the problem of navigation and escape.

After stating the characteristics and elements of the problem that is solved through the designs presented in this thesis, the next section includes an overview of the main decisions made on the considerations, assumptions and approaches to solve the problem of navigation and escape.

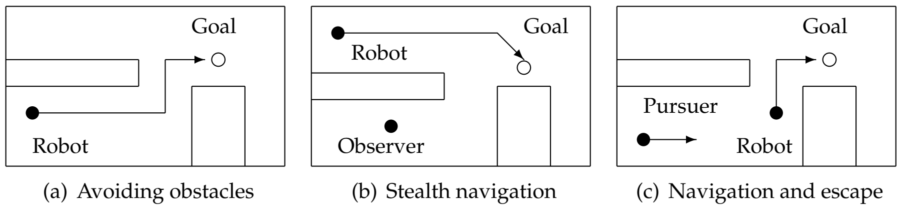


Figure 4.1: The problem of avoiding obstacles, a stealth navigation problem and the problem of navigation and escape from pursuers.

4.2 Approach to the solution of the problem

Many different alternatives are available to solve the problem of navigation and escape from pursuers, coming from a broad range of subjects including control systems and computer science. This section discusses the considered options and the decisions made regarding:

- The type of control approach.
- The main considerations for the problem: the characteristics and constraints of the environment and the proposed solutions.
- The particulars of the designs, including which techniques and algorithms are suitable to solve the problem, and how to adapt them and assess their success.

The problem of navigation and escape can be considered a planning problem, but also a game or a control law problem. It depends on the specifics of the desired solution: high level or low level, for a particular physical configuration of a robot or for specific processing and sensing capabilities. As not all the possible combinations can be explored in a short period of time, two main approaches for the solution have been proposed, from path planning and game theory with reinforcement learning. Both try to fulfil the main objectives of flexibility and universality for the implementation in different robotic platforms, and computing solutions with available resources online.

4.2.1 Selecting a control approach: reactive or deliberative

Two main kinds of general architectures, reactive and deliberative (Chen et al., 2008), were considered for the approach to the design of the control system, both illustrated and exemplified in Figure 3.6. A deliberative high level approach is more suitable for the solution of the problem of navigation and escape for different robotic platforms, as a high level control system can be later coupled to a low level control system that characterises the dynamic and kinematic properties of every robot (Kim and Shim, 2003). The deliberative approach for the designs accomplishes the objective of universality of the designs, mentioned in the introductory chapter. Also, planning motions in advance is advantageous to achieve hiding, disorienting the pursuers and paths that lead to specific locations, whereas reacting to instantaneous information could lead to being herd towards a location, or merely randomised navigation.

4.2.2 Considerations about the environment

Elements in a real life environment are continuous in geometry and motion. When continuous environments are processed through sensors, a discrete representation of the environment in space and time is computed from the extracted information that can be approximated as continuous, or kept as discrete. Thus planning can be performed over continuous or discrete environments (in both spatial and temporal dimensions).

Choosing between a continuous or a discrete model of the environment involves a trade-off between accuracy combined with the amount of information available, and memory capabilities (for processing and storing information). Using real life robotic platforms with limited sensing capabilities eliminates the problem of dealing with large amounts of data, and shifts it to the problem of succeeding in a task with limited information about it. Considering an environment representation that comes from sparse sensed points, the work presented in this thesis computes discrete navigation paths, that can be converted into a continuous motion by coupling a lower level control system (e.g., a controller to go from a point in the path to the next in a smooth motion).

Many previously proposed designs in path planning consider fully visible, previously explored and mapped environments. But this assumption of previous information is not realistic when robots are deployed in dynamic unknown environments, hence the need to extend the process of design to work with limited information and generate solutions accordingly. In the work presented in this thesis, the robot does not know anything about the environment but it can build a local map or representation from sensed information. The robot computes a map of the observed entities in the environment to decide what to do to escape from the pursuers if they are in direct reach. The map is incomplete in a continuous sense, as the sensors have a limited range, and some spaces cannot be observed behind the obstacles or behind the pursuers. The map is not necessarily stored in memory, as generating maps of the environment has not been considered strictly part of the solution to the problem of navigation and escape. Therefore, the sensed map needs to be updated as the robot moves around.

Also, generating designs that can be adapted to different kinds of sensors has been part of the objectives in the work of the thesis. In this manner, solutions can be computed using the same proposed mechanisms, but with different ranges of visibility and certainty of the information from the environment.

4.2.3 Choosing an approach to solve the problem

The most crucial step towards the design of solutions to the problem of navigation and escape is the selection of particular approaches. A wide range of options can be found in the fields of artificial intelligence from a computational perspective and control systems. Three interesting possibilities have been selected, due to their significance in the study of relations between pursuers and evaders, and other related problems like stealth navigation. It is important to mention that most of the current research regarding the relations between pursuers and evaders does not attempt to design intelligent evaders, and in the cases where individuals are optimally designed, the main focus is on the pursuers, as mentioned in the previous chapter. The three possibilities are described in the next paragraphs, followed by a general methodology for the overall design of solutions.

Solutions from a path planning perspective Computing paths for collision avoidance is a very broad area of research. Many specific algorithms have been generated to deal with different characteristics of the navigation: dynamic and static obstacles, avoiding other robots, coverage of an area, including a mapping or self-localisation while navigating, covert navigation, reactive navigation, to mention some. The main techniques used in path planning have been grouped in Section 3.1.

Although some research has been done for covert or stealth navigation, many of these approaches and others found for the problem of evasion in cluttered environments use a potential function approach (Ravela et al., 1994; Masoud, 2002; Birgersson et al., 2003; Masoud, 2003a,b; Tews et al., 2004a,b). The advantages of potential functions over other methods in terms of intuitive understanding of the main ideas, an easier computational implementation and its success for sensing based navigation in dynamic environments using different robotic platforms, dynamic and physical constraints, and terrain, make extending the use of potential functions for the specific problem of navigation and escape a viable solution. Also, the use of potential functions does not require a complex analysis of the environment, compared to visibility graphs or Voronoi diagrams. Reviewed literature demonstrates an occupancy grid is enough to obtain a navigable path (Koren and Borenstein, 1991; Borenstein and Koren, 1991; Tsourveloudis et al., 2001; Wang et al., 2002; Birgersson et al., 2003; Shem et al., 2008; Cherubini and Chaumette, 2010; Zhang and Gao, 2011). For all these reasons, a path planning approach based on potential functions has been considered in the computation of solutions to the problem of navigation and escape.

Solutions from a game theory perspective Pursuit-evasion games have been used in many occasions to design control systems that effectuate individual and collective pursuit (in a cooperative setting in the case of multi-agent systems). Evaders are traditionally not studied or developed as much as pursuers from a design perspective, which does not correspond to nature and real life situations, where animals and humans are capable of deciding and using different strategies to achieve escape from pursuit. Studying the evasion perspective offers new challenges for the design of more efficient pursuers. The use of reinforcement learning in the solution of pursuit-evasion tasks allows the computation of navigation strategies based on rewards or punishments given chosen actions, useful when no model of the opponent (pursuers) is available.

Solutions based on inspiration from biology The increasingly strong and popular area of bio-robotics has demonstrated that biological natural processes are a useful source of inspiration to create innovative solutions to problems, using ideas and mechanisms that have been produced after long periods of evolution and might be more efficient for a particular task. Biologically inspired features from anti-predator behaviours have been considered in this thesis to produce a solution to the problem of navigation and escape, making a robot act in a similar manner to a real prey. The ideas surrounding the biological inspired features used in the designs are expanded in Section 4.3.

Methodology for the design of the solutions

A methodology for the designs presented in this thesis has been proposed. The design of any system that allows obtaining a solution to the problem of navigation and escape from pursuers has been divided into two main parts, inspired by the procedure for designing planning algorithms by Hwang and Ahuja (1992a):

1. How to process the environment, based on the considerations explained in Subsection 4.2.2 and specifically for the selected approaches of path planning with potential functions and pursuit-evasion games solved by reinforcement learning.
2. How to choose the actions to follow to fulfil the objectives of navigation and escape, also based on path planning with potential functions and pursuit-evasion games solved by reinforcement learning.

The next subsection presents a further exploration of the decisions regarding the particulars of the path planning approach, including the use of potential functions and a discrete environment representation. The selection of particulars for the game theory approach, including the use of reinforcement learning for the solution of pursuit-evasion games from the evasion perspective, is detailed in Chapter 7.

4.2.4 Path planning with potential functions and discrete environment

An optimal or suboptimal global path planning approach involves obtaining the best possible path according to the measurement of optimality and other constraints like time, for the whole problem. But real life navigation could involve dealing with local portions of the dynamic environment from the sensors (Zhu, 1991), where a navigation path is formed by local solutions corresponding to the current knowledge of the environment. Joining these local solutions, which could be optimal or suboptimal, would not necessarily result in exactly the same optimal global solutions if the whole environment was known before the deliberation took place. The designs proposed in this thesis are intended to solve the problem of navigation and escape from pursuers for sensed local dynamic environments.

Following the methodology, the first step was selecting a representation of the environment, based on potential functions. The representation of the environment needs to cover the following requirements, considering environment information from sensors and potential functions as the main path planning method:

- The representation should support sensed, continuous, local, dynamic environments, providing a practical and computationally inexpensive map of the entities in the environment.
- The representation should make possible the incorporation of bio-inspired features to improve proposed solutions to the problem of navigation and escape from pursuers.
- The representation needs to make possible the computation of a path that avoids obstacles whilst also avoiding capture by pursuers, using a potential function based approach, thus preventing local minima and any other situations that would prevent computing a path.

Many different kinds of environment representations have been used in path planning, all of them closely related to the information available, and other simultaneous task goals like self localisation, coverage or mapping. Environments can be represented by polygons, neural networks, surfaces, grids, topologic maps, hidden Markov models, and many other options. In the stated problem of navigation and escape of this thesis, obtaining a complete or accurate stored model of the environment is not contemplated, but only a representation of the environment that allows computing a solution to the problem, as mentioned before.

Some of the considered representations for the designs were a neural network, polygons and cell decomposition (exact or approximate). Some sensing mechanisms can only identify sections of the obstacles, limiting the available information to discrete samples in the space of the environment. Most of the methods in path planning that deal with known and unknown environments suggest some form of discretisation or sampling to represent continuous environments and continuous robot configurations, as environments are not necessarily discrete in origin (Bruce and Veloso, 2002; Choset et al., 2005; Marzouqi and Jarvis, 2006). In order to obtain useful and fast representations from samples of the environment provided by a sensing stage, a grid with exact or approximate cell decomposition is a feasible option.

Occupancy grids and other grid representations have been used to solve the problem of autonomous navigation, including in combination with potential function based path planning methods. They provide a reduction from a continuous model to samples of configurations. This is particularly useful when a large number of configurations and information from the elements of the environment is available, but processing needs to be done in a limited amount of time to provide a solution to autonomous navigation, and also when only samples of configurations or information are available or required to be used. Although the information of the environment from sensors might be discrete, navigation in the real world is a continuous process. But computing a path from a discrete representation (in this case a grid) does not necessarily lead to the production of a discrete navigation implementation, as interpolation between nodes in a discrete path can be done in a low level tracking control or in a more reactive layer (Wang et al., 2002; Huh et al., 2002).

Another advantage of mapping the environment using an approximate cell decomposition representation is its widespread use in conjunction with potential functions. The functions mentioned in Subsection 3.3.1 can be calculated for sampled points in the environment (including the gradient of the functions), to compute a path that connects the gradient descent of the functions towards a goal. Subsection 3.3.2 shows the computation of scalar potential fields based on a boundary value problem (BVP) for harmonic functions, where an approximated solution to the BVP can be achieved if approximated to a grid and then using a numerical method. Grid approximated potential fields save computation time when it is crucial for the success of the task, as it is in the case of navigation and escape, where moving away from the pursuers is vital for the robot in question. Vector potential field representations can be used in a discrete way, as scalar fields, if samples of the vectors in some points of the environment are taken and used to compute a path.

As mentioned previously, artificial potential functions are a path planning mechanism to map the environment that deals with the requirements of flexibility (for adaptation to different sensorial platforms), versatility (i.e., the same map can be used with different algorithms like search for path planning, or reinforcement learning as a reward function), adaptability (i.e. being easily updated if the environment changes), and easy implementation.

From all the literature reviewed in Chapter 3 concerning the different forms to compute potential functions, and a discrete grid based representation, the following particular options were considered for the designs:

- Popular quadratic attraction and repulsion functions (Equations 3.2 and 3.4, respectively). The gradient can be computed mathematically.
- Computing an approximation of BVP with harmonic functions and different boundary conditions, from numerical methods like SOR. These functions guarantee the absence of local minima mathematically.
- Wave-front expansions, that contain only one global optimum value.

Experimentation allows comparing the different configurations used for the potential functions, in terms of their extensibility to the problem of navigation and escape from pursuers. Pursuers need to be incorporated into the potential functions, as well as the bio-inspired features, and not all the potential function implementations might be up to the requirements.

The processing of the environment is studied later in this chapter, followed by the computation of a path, along with examples to illustrate the proposals. The evaluation of results from simulations when pursuers are simulated simultaneously is presented in Chapter 5.

The second step in the methodology mentioned in Subsection 4.2.3 is the design of the computation of a path, i.e. how to select the navigation actions that guide the robot to avoid collisions with obstacles and being captured (reached) by pursuers. The computation of a path depends on the representation of the environment, as the latter provides the necessary information. A potential field from discretised potential functions would provide a discrete set of magnitudes or vectors to compute a path from an initial position to the goal avoiding obstacles, by connecting the available cells or samples following the minimisation of the magnitude or the direction of the resulting forces.

Steepest descent, a discrete version of gradient descent (Johnson and Picton, 1994), has been commonly used to compute a path for potential functions. Nevertheless, if local minima are present, steepest descent might not compute a path that leads to the desired navigation goal. Adding other heuristics and different search methods have been proposed, as mentioned in Section 3.3.6, to solve the problem of local minima due to the geometry of the obstacles and the complexity of the environment. When escaping from pursuers in a sensed local dynamic environment, heuristics differ from those of simple navigation in previously known environments, as priorities change and navigation goals might do. Following the steepest descent is enough if the design of the potential functions does not give any margin for local minima, and if the heuristics are formulated in a way that is compatible with the search method. Otherwise, a different implementation of a search method is needed according to the proposed heuristics.

Other added features in the path planning include an updating of the path or navigation strategy according to the availability of the information from sensors in a dynamic environment, and using probability maps to identify unknown parts of the environment. The first feature allows dealing with a completely unknown dynamic environment, and eliminates the need of storing a map of the whole environment. Using probabilities to represent uncertainty is a more realistic approach for stochastic situations, as what lies behind the visible parts of the environment cannot be predicted accurately. In order to fulfil a navigation and escape task with dynamic pursuers in a dynamic environment, the planning has to be updated online whilst the actions take place, as the restrictions and the suitable goal may change.

Finally, it is important to clarify that the purpose of the high level control system is not necessarily optimal in terms of shortest distances to a goal, as maximising the chances of survival for a robot could involve following suboptimal or randomised navigation strategies.

4.2.5 Modelling the pursuers

For the purposes of the thesis, it is assumed that pursuers have explored the environment previously, but at the same time they have limited sensing capabilities regarding their own surroundings. It is also assumed that they can communicate to each other the detection of a robot.

Pursuers are necessary as an objective of the designs presented in this thesis is to create a control system that achieves navigation and escape from them, regardless of the kind of pursuer that is encountered in the environment. Two main kinds of pursuers have been considered for the designs of an escaping robot, from the literature review:

1. Pursuers that follow the evader, trying to shorten their distance to the evader at all times.

2. Pursuers that try to predict the evader according to their observations, and then move towards that location, i.e., interception tasks (Anderson and McOwan, 2003; Glendinning, 2004; Justh and Krishnaprasad, 2006; Park et al., 2009, 2010).

4.2.6 Simulations and implementation

Graphic simulations in MATLAB have been proposed and implemented to validate the proposed designs through statistics. The different potential functions, along with the computation of paths incorporating the bio-inspired features, are compared and evaluated under different configuration of obstacles and pursuers.

No model of the dynamics of the robot is assumed, but a perfect translation and rotation of coordinates, as the high level designs are intended for any physical platform, being coupled by a low level control system that approximates the path according to the physical and dynamic constraints of the robot. But the simulations consider different velocities of navigation for both the robot and the pursuers.

4.3 Bio-inspiration for navigation and escape

The following ideas, directly inspired by biological behaviours, have been used to solve the problem of navigation and escape from pursuers:

Innate behaviours A system that performs actions of escape from the first interaction with pursuers in the life of the robot is needed, to provide an equivalence to innate behaviours in animals and representing the self-survival capabilities of real live beings. Escaping has been chosen as the response to pursuit over any kind of counter-attack as it is frequently observed in nature.

Responding to pursuers An action needs to be performed to escape pursuers as soon as pursuers are detected in the environment (responsive actions) or if pursuers are likely to exist in the environment (prevention actions). Broom and Ruxton (2005) proposed a model for the start of the chase-evasion motions according to the distances of detection and the opponent change of mode from normal movement to chase or evasion. But the robot might not have any previous information on the maximum velocity nor the intelligence capabilities of the pursuers due to the lack of previous experience, thus an immediate action that maximises the chances of the robot to avoid capture is needed.

Hideaways as navigation goals Literature on anti-predator individual behaviours reviewed in Chapter 2 indicates that animals use refuges to prevent predation, providing an inspiration to design temporary refuges behind obstacles as a possible temporary navigation goal for the problem of navigation and escape (Caro, 2005). A hideaway (or refuge) is a blockage in the direct view between the pursuers and the robot, and it is a temporary solution to the problem of pursuit. When the navigation has a global goal or location that the robot needs to reach, using different hideaways along the path becomes a covert or stealth navigation.

A hideaway in a continuous 2-D location (x, y) has been proposed in this thesis as a function of the line of sight of pursuers, the certainty in the knowledge of the environment, the distance to the robot, the distances to the pursuers, and the distance to the global navigation goal if it exists. Then,

$$h(x, y) = f(\text{sight, certainty}, d(h(x, y), \text{robot}), d(h(x, y), p), d(h(x, y), \text{goal})) \tag{4.1}$$

where a measure of the sight of the pursuers and the certainty of their information are included, $d(h(x, y), \text{robot})$ is the distance from the hideaway to the robot, $d(h(x, y), p)$ is the distance between the hideaway and the $p = 1, ..., n_p$ pursuers, and $d(h(x, y), \text{goal})$ is the distance between the hideaways and the navigation goal. The computation of hideaways can be performed as part of the analysis of the environment, considering the corresponding grid of elements observed by the robot. The hideaways as temporary goals should be incorporated then into the computation of the potential functions.

Proteanism in the navigation Proteanism can be observed in different species as a natural anti-predator behaviour (Caro, 2005), as mentioned in Subsection 2.2.1. It provides a tool to improve the escape against intelligent pursuers that are able to model the navigation of a simple escaping robot (e.g., long linear motions), by incorporating a degree of randomness. The randomness in the navigation increases the complexity of a simple prey reaction like running in the same line of motion as the pursuer, where other pursuers that know this could intercept the robot, as illustrated in Figure 2.3. Protean navigation also provides a change in the perspective of the environment, that serves to discover new regions through the sensing that could help to the escape.

Furuichi (2002) proposed a model of chase-evasion motions of two animals in a two dimensional setting that considers vector forces to indicate the direction and magnitude of displacement (Subsection 2.2.3 and Figure 2.5). Proteanism has been incorporated as a random unitary vector that modifies the direction of motion and creates circular patterns when the pursuer is close (Furuichi, 2002). Also, many co-evolutionary studies have shown proteanism as a result of the interactions between two species (Miller and Cliff, 1994; Miller, 1997). Co-evolution is computationally expensive to compute online optimal strategies for navigation and evasion. Adaptive control systems are more desirable than others calculated offline, as the environment could change abruptly and the previous parameters could not be valid anymore.

Proteanism in a specific continuous 2-D location, (x, y), is proposed as a function of the distances between the location and the pursuers, $p = 1, ..., n_p$, the distance between the location and a local goal of navigation for the sensed local part of the environment, the energy of the robot, and other characteristics of the pursuers like velocity. Thus

$$\text{proteanism}(x, y) = f(d((x, y), p), d((x, y), LG(x, y)), \text{energy}, \text{others}_p) \tag{4.2}$$

where $d((x, y), p)$ is the distance between a location and the pursuers, $d((x, y), LG(x, y))$ is the distance between a location and a local navigation goal, and the other parameters are the same as for the hideaways equation. A measure of the energy of the robot is included in the equation, along with a measure of other parameters and characteristics of the pursuers, others$_p$.

Two forms of incorporating the proteanism into the designs are proposed here: (a) *a priori* within the representation of the environment, and (b) *a posteriori* once the optimal or suboptimal path has been computed. Proteanism can be incorporated *a priori* as subgoals between the current location and the local goal (guided proteanism), or guiding the robot to a temporary random location if there is no other navigation goal, in the computation of scalar potential functions. The subgoals can be interpreted as reaction forces valid for intervals of time (Furuichi, 2002). The idea of incorporating subgoals into the potential functions has been previously proposed as a solution for local minima problems (Masoud, 2003a; Jing and Wang, 2004; Bell, 2005; Weir et al., 2006; Cen et al., 2007; Tan et al., 2010), but their usage, design and computation have a new meaning in this thesis.

General structure of the designs All the proposed designs follow the process for planning algorithms by Hwang and Ahuja (1992a), mentioned previously: how to process the environment first, and then how to deliberate the actions, both with added biologically inspired features. Figure 4.2 shows a schematic of the components drawn in both desig steps, incorporating the bio-inspired features from Section 4.3. The dashed lines correspond to the variations in the designs, including the use of proteanism *a posteriori* to enhance a discrete path.

The next section describes the proposed analysis of the environment, which computes a discrete representation of a sensed environment, evaluates the elements, chooses a local goal from bio-inspired hideaways and distances, chooses subgoals for a protean navigation, and results in potential functions that are used to compute an adequate path.

4.4 Analysis of the environment

The analysis of the environment involves computing a suitable representation from the available information. As stated before, the information is provided by sensors in a discrete form, where the elements of the environment are identified by an approximate cell decomposition. Using samples from sensors reduces the amount of data to process for continuous environments (Bruce and Veloso, 2002), and if sparse discrete sensing is used to identify the elements in the environment, sampling is obliged. Cell decompositions (grids) proportionate a representation of the environment that can be used in conjunction with potential functions, to translate obstacles and pursuers into sets of points (interpreted as

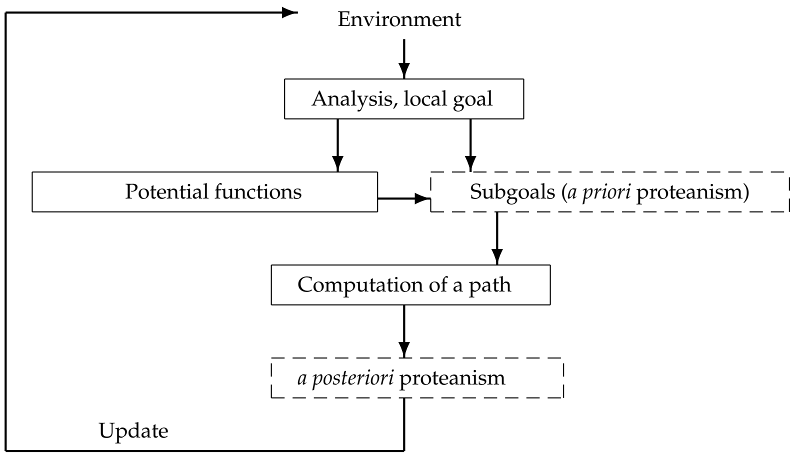


Figure 4.2: Schematic of the elements in the proposed designs with bio-inspired features. The dashed lines represent variations in the proposals compared to the methodology by Hwang and Ahuja (1992a).

the centroids of the cells in a grid). Then, respective potential functions can be used to compute a navigation strategy, as done by Luo and Yang (2008).

The proposed procedure to analyse the environment includes a series of steps, detailed and exemplified in the following subsections:

1. Approximating the environment as a grid, where elements are identified and approximated to cells according to the dimensions of the robot (Subsection 4.4.1). For design purposes, it is assumed that the robot has general innate pursuer recognition, as young animals do when they are born and have not had the chance to learn.
2. Computing the possible hideaways (bio-inspired refuges) as shadows considering all the cells in the environment (Subsection 4.4.2).
3. Computing a measure of the distances between any possible free space cell and the pursuers, and free cells and the position of the robot (Subsection 4.4.3).
4. Computing a total function in terms of the distances and the hideaways that allows choosing a local navigation goal (Subsection 4.4.4).
5. Computing bio-inspired proteanism through subgoals, considering the local navigation goal and the identification of elements (Subsection 4.4.5).
6. Computing the corresponding potential functions (Section4.5) for the subsequent path planning.

4.4.1 Approximation of sensed elements into a grid

A grid is built with cells of dimensions $n \times n$, where n is the longest dimension of a robot (length or width) in a 2-D plane. This selection of cell size allows sampling possible navigation configurations of the robot considering its dimensions. The total number of cells in the grid depends on the range of visibility or sensing of the robot, and cells that are added out of the range of sensing to predict probable elements and more probable free space. The grid represents a temporal and spatial discretisation of the planning task, as the environment is sampled in a discrete instant of time, and represented in a discrete manner.

Obstacles and pursuers are approximated within the chosen size of the cells (i.e., if any part of the obstacle or pursuer is contained in a cell, the cell becomes part of the obstacle or pursuer), to favour the avoidance of collisions and the elimination of ravine problems. Any cell in the grid can be an obstacle, a pursuer, a known free space, or unknown (out of the reach of the sensors or occluded behind obstacles and pursuers). It has been assumed that: *(a)* the robot is able to differentiate and recognise pursuers and obstacles; and *(b)* the robot is a cell in the centre of the grid of its own sensing when its view is of 360 degrees. The robot is allowed to move from cell to cell if they are contained in the free space, i.e. not occupied by probable obstacles and pursuers.

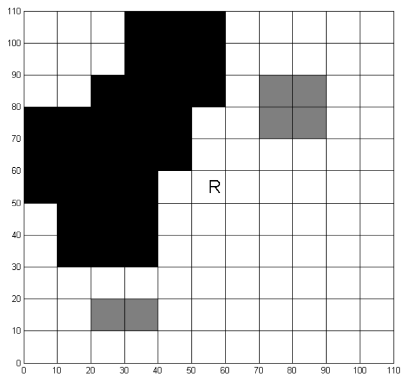
Probabilities are assigned to every cell in the grid according to the sensed information. A cell with centroid in (a, b) has all of the following probabilities:

$$P_{(a,b)} = \begin{cases} P_{(a,b)}(p) \ \in [0,1] & \text{of being a pursuer} \\ P_{(a,b)}(o) \ \in [0,1] & \text{of being an obstacle} \\ P_{(a,b)}(F_s) \ \in [0,1] & \text{of being free space} \end{cases} \tag{4.3}$$

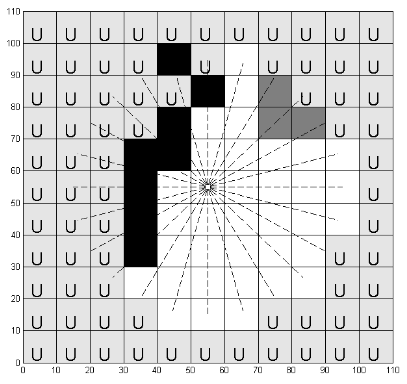
where the probabilities $P_{(a,b)}(p)$ and $P_{(a,b)}(o)$ are mutually exclusive, thus $P_{(a,b)}(F_s) = 1 - P_{(a,b)}(p)$ or $P_{(a,b)}(F_s) = (1 - P_{(a,b)}(o))$ correspondingly. This means an obstacle cannot be a pursuer, nor a pursuer can be an obstacle. The probability of being free space gets smaller if the cell is probably an obstacle or a pursuer. If the identification of an element is considered to be accurate, $P_{(a,b)}(p) = 1$ if the cell is part of a pursuer, $P_{(a,b)}(o) = 1$ if the cell is part of an obstacle or $P_{(a,b)}(F_s) = 1$ if the cell belongs to the free space.

In order to determine the probabilities in the unknown cells, it is possible to perform some predictions about the environment. For example, the adjacent cells behind obstacles and pursuers are more likely to be part of the obstacles and pursuers than more distant cells.

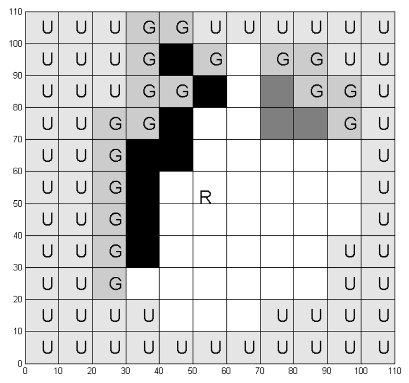
Figure 4.3 shows an example of the discrete analysis of the environment using a grid of 11×11 cells, where the range of sensing is approximately $3 \cdot$ size cells. The approximated discrete environment contains a maximum of 5 randomly placed square obstacles with a maximum size of 9 cells each, and a maximum of 2 randomly placed pursuers with a maximum size of 4 cells each. A probability of $P_{(a,b)}(o) = 0.5$ or $P_{(a,b)}(p) = 0.5$ has been assigned to unknown cells behind obstacles and pursuers, respectively, and the resulting probability of the guessed cells of being free space is also 0.5, considering that if $P_{(a,b)}(p) > 0$ then $P_{(a,b)}(o) = 0$, or if $P_{(a,b)}(o) > 0$ then $P_{(a,b)}(p) = 0$. This means that a cell cannot probably be an obstacle and a pursuer simultaneously in the example. The sensing in the example has an error of 1% detecting the elements in the environment, due to parameters in the algorithm.



(a) Environment



(b) Sensing and identification



(c) Extension of elements

Figure 4.3: Randomly generated discrete environment, the representation of the sensing and identification of elements, and extension of the identified obstacles and pursuers to the unknown cells by assignation of probabilities to the adjacent cells behind the elements. Obstacles in black, pursuers in grey, free space in white, unknown in light grey with a U, extended pursuers and obstacles cells in grey with a G, and the robot in the centre.

4.4.2 Computing bio-inspired hideaways

The inspiration behind the idea of hideaways has been explained in Section 4.3. Hideaways are the shadows behind the obstacles and out of the line of vision or sensing of the pursuers physically. It is assumed that a robot in a hideaway cannot be seen by the pursuers with occluded vision line, and therefore reaching that location is a temporary solution to the problem of navigation and escape.

A cell can be a hideaway if it lies behind an obstacle and it is out of the line of vision of a pursuer. The best hideaways are the ones that intersect for all the pursuers in the same location, as that would mean none of the visible pursuers are able to see that part of the space. The computed hideaways change according to the motions of the robot and pursuers, thus they need to be recomputed along with the environment representation to obtain a path that suits the current environment state.

Considering the equation presented in Section 4.3, a hideaway can be defined as

$$h(x,y) = f(\text{sight, certainty}, d(h(x,y), \text{robot}), d(h(x,y), p), d(h(x,y), \text{goal})) \tag{4.4}$$

where the sight can be computed as a value that indicates the amount of intersection of the shadows for all the pursuers, $w_h(a,b)$ for all the cells in the environment, (a,b). The certainty of the information is given by $P_{(a,b)}(F_s)$, to consider free space cells as the hideaways. If the navigation has no global goal, then

$$h(a,b) = f(w_h(a,b), P_{(a,b)}, d((a,b), \text{robot}), d((a,b), p)) \tag{4.5}$$

where a cell is a hideaway according to the computed shadows and the distances between a cell and the robot, $d((a,b), \text{robot})$, and between a cell and the pursuers, $d((a,b), p)$.

The value $w_h(a,b)$ of each cell can be calculated using the proposed procedure:

1. Calculating the angle between a probable obstacle cell and a probable pursuer p (computing the centroid of the pursuer according to the respective probable pursuer cells).
2. Locating the angle in one of eight sectors, or the vertical and horizontal lines, as shown in Figure 4.4. The shapes of the shadows cast by an obstacle from the perspective of a pursuer have been approximated by triangles shown in Figure 4.4 or lines, a set of cells in the discrete environment.
3. The shadows behind all obstacles (k_o probable obstacle cells) are computed for all the n_p pursuers, and their intersections are accumulated, so that every cell in the grid that is not an obstacle or a pursuer has a value of

$$w_h(a,b) = \begin{cases} 1 & \text{if intersects } n_p \text{ pursuers} \\ \frac{p}{n_p} & \text{if intersects for } p \text{ pursuers} \\ 0 & \text{if never a shadow} \end{cases} \tag{4.6}$$

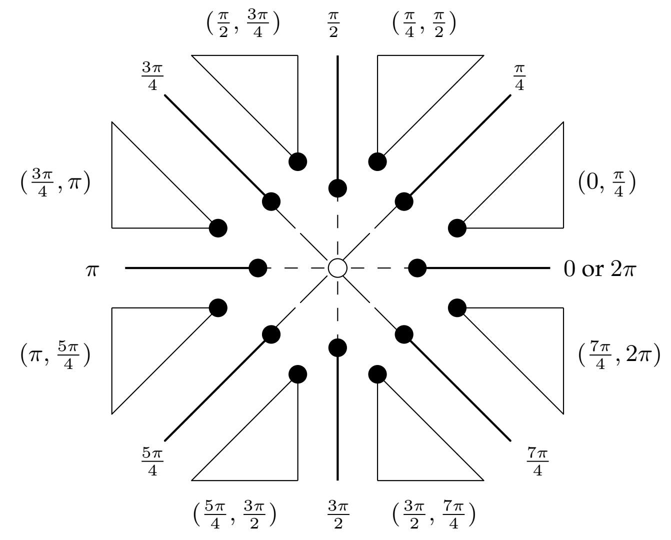


Figure 4.4: Division of the environment in sectors according to the angle between a pursuer (represented by a white circle) and obstacles (represented by black circles), and approximated corresponding triangular or linear shadows.

The cells with $\max w_h(a, b)$ are the best hideaways, $h^*(a, b)$, as they are out of the sight of more pursuers in the sensed environment. An example of the computation of $w_h(a, b)$ for the environment in Figure 4.3(a) is shown in Figure 4.6(a).

4.4.3 Computing distance functions

As part of Equation 4.5 regarding the distances, functions of the distances between pursuers (centroids of probable pursuers) and free cells, and distances between the robot (in the centre of the grid) and free cells are computed for every cell (from their centroid) using the proposed equation:

$$w_d(a, b) = g\left(d((a, b), \text{robot})\right) \cdot \prod_{p=1}^{n_p} f(d((a, b), p)) \tag{4.7}$$

where $g\left(d((a, b), \text{robot})\right)$ is a function of the Euclidean distance between the robot (in the centre of the grid of its own sensing) and a free cell, and $f(d((a, b), p))$ is a function of the Euclidean distances between a pursuer p and a free cell for all n_p pursuers. The functions f and g can have any shape, but they are distributed radially within all the points in space. Linear functions have been used for the simulations presented in this thesis, as shown in Figure 4.5:

$$f(d((a, b), p)) = \frac{1}{\sqrt{2}D_{grid}} d((a, b), p) \tag{4.8}$$

$$g\left(d((a, b), \text{robot})\right) = -\frac{d((a, b), \text{robot})}{\sqrt{2}D_{grid}} + 1 \tag{4.9}$$

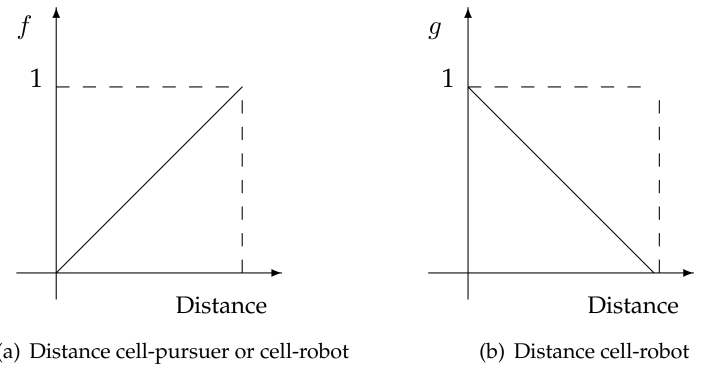


Figure 4.5: Linear functions for the distance between a cell and a pursuer or a cell and the robot (if no pursuers are detected), and for the distance between a cell and the robot in the presence of pursuers.

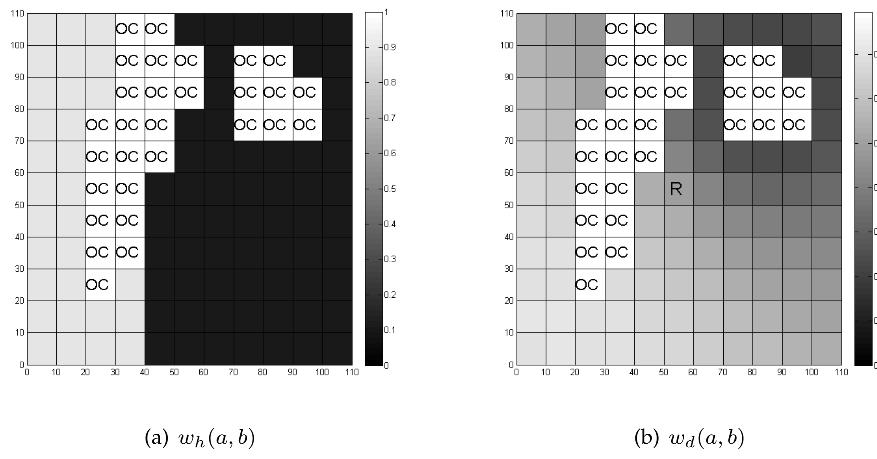


Figure 4.6: Computed values for the shadows in every cell, $w_h(a,b)$, for the environment in Figure 4.3(a), and the function of the distances, $w_d(a,b)$, for every cell. Probable obstacles and pursuers indicated with OC, and the robot in the centre.

where D_{grid} is the width of the grid, $d((a,b),p)$ is the Euclidean distance between a pursuer and any free cell, and $d((a,b),\text{robot})$ between the robot and any free cell, respectively.

The function $w_d(a,b)$ has large values for the cells that are far from pursuers and closer to the robot. Its maximum is the best location far from pursuers. If no pursuers are sensed in the environment at a determined time, the function only quantifies the distances between cells and the robot, and the maximum is a free cell in the edge of the observed environment to favour exploration,

$$w_d(a,b) = \begin{cases} g\left(d((a,b),\text{robot})\right) \cdot \prod_{p=1}^{n_p} f(d((a,b),p)) & \text{if pursuers} \\ f\left(d((a,b),\text{robot})\right) & \text{if no pursuers} \end{cases} \quad (4.10)$$

An example of the computation of $w_d(a,b)$ is presented in Figure 4.6(b), for the randomly generated environment of Figure 4.3(a), and the functions f and g.

4.4.4 Selecting a local navigation goal

The local goal of navigation, LG, is a planned location to reach according to the perceived information in the moment of sensing. If hideaways can be computed, the robot should try to reach the best one. Otherwise, a location far away from the pursuers is chosen as a temporary solution. Then,

$$LG = \begin{cases} h^*(a,b) & \text{if } h(a,b) \neq \emptyset \\ w_d^*(a,b) & \text{otherwise} \end{cases} \tag{4.11}$$

where

$$h^*(a,b) = \max_{\forall(a,b)} \big(w_h(a,b) \cdot P_{(a,b)}(F_s) \cdot w_d(a,b)\big) \tag{4.12}$$

is the best hideaway, as explained in Subsection 4.4.2; and

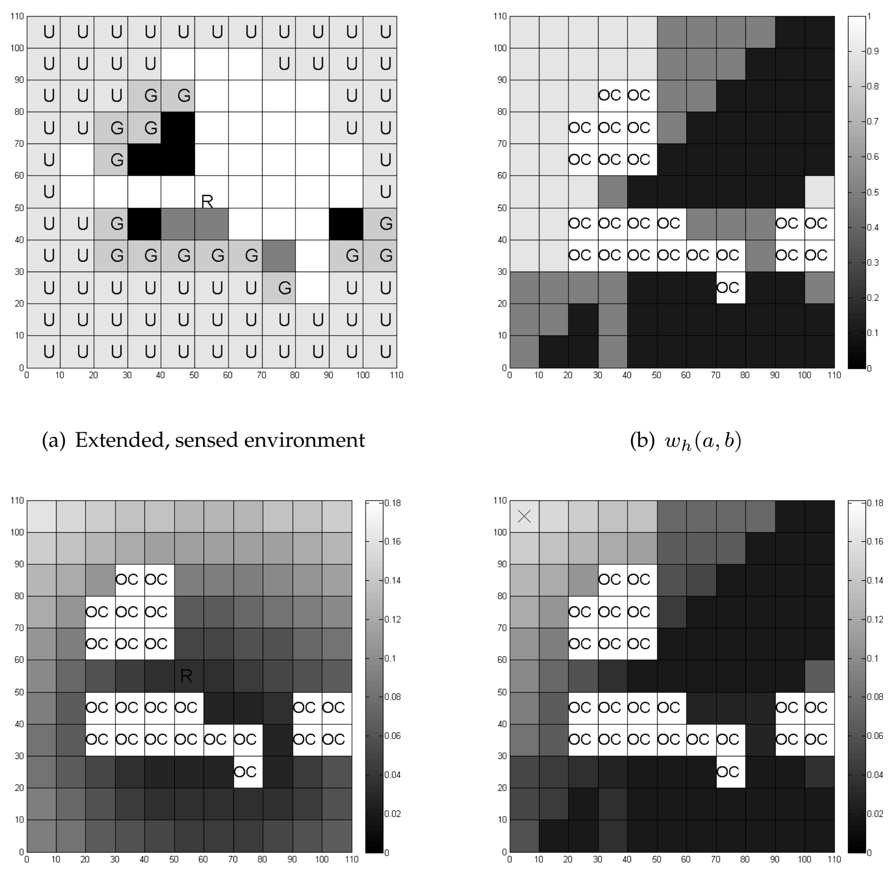
$$w_d^*(a,b) = \max_{\forall(a,b)} w_d(a,b) \tag{4.13}$$

is a location that leads far from the pursuers as a function of the distances between pursuers and the cells, and the cells and the robot as explained in Subsection 4.4.3. An example of the computation of a local goal from the hideaways is shown in Figure 4.7, for a randomly generated discrete environment of 11×11 cells, with a maximum of 5 randomly placed obstacles with a maximum size of 9 cells each, and a maximum of 2 randomly placed pursuers with a maximum size of 4 cells each. A sensing range of $3 \cdot$ size cell has been used to identify the elements in the environment, with a probability of 0.5 for extended pursuers and obstacles. Another example of the computation of a local goal when no hideaways are available is shown in Figure 4.8 for a similarly generated random discrete environment.

4.4.5 Bio-inspired *a priori* guided proteanism through subgoals

As part of the bio-inspired features added to the designs of solutions to the problem of navigation and escape from pursuers, proteanism has been incorporated to the navigation to provide randomness in the path that could cause confusion to the pursuers, and would improve local minima problems. Proteanism is used when pursuers are detected in the environment, as collision avoidance does not solve the problem of escape from pursuit.

The use of guided subgoals is inspired by anti-predator proteanism behaviours (from Caro (2005); Furuichi (2002); Nolfi and Floreano (1999)), and subgoal based navigation (e.g., in Bell (2005)). The robot chooses a subgoal, and then navigates towards it. Then, another subgoal is chosen, and the robot moves towards it, repeating this process until the robot has reached a hideaway or local goal, or the pursuers have desisted in their chase after some time. The produced motion plan is not optimal in terms of taking the shortest distance towards a goal, but it is intended to enhance the possibilities of survival of the robot in the environment. The proposed implementation of the guided subgoals is equivalent to the random direction change in Furuichi (2002), but the proteanism is oriented towards the local goal, according to the elements in the environment.



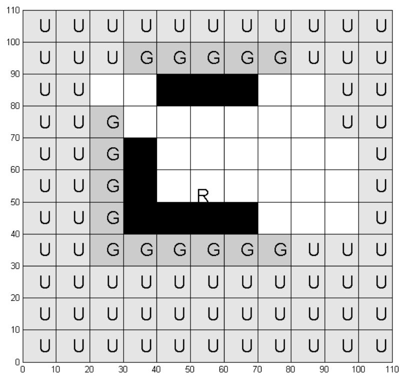
(a) Extended, sensed environment

(b) $w_h(a, b)$

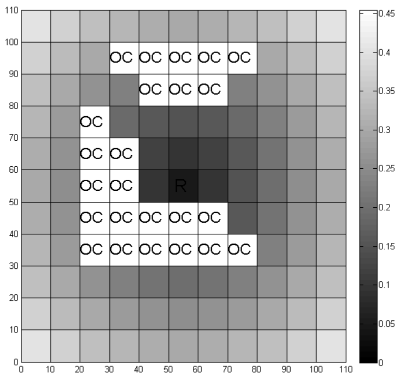
(c) $w_d(a, b)$

(d) $h^*(a, b)$

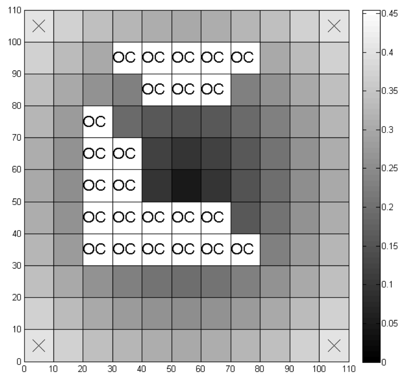
Figure 4.7: Computed $w_h(a, b)$, $w_d(a, b)$ and LG as $h^*(a, b)$ (probable obstacle and pursuer cells indicated with OC and the LG as X) for an extended, sensed, randomly generated environment (obstacles in black, pursuers in grey, free space in white, unknown in light grey with a U, extended pursuer and obstacle cells in grey with a G, and the robot in the centre).



(a) Extended, sensed environment



(b) $w_d(a,b)$



(c) $w_d^*(a,b)$

Figure 4.8: Computed $w_d(a,b)$ snf LG as $w_d^*(a,b)$ (probable obstacle and pursuer cells indicated with OC and equidistant local goals as X), for an extended, sensed, randomly generated environment (obstacles in black, pursuers in grey, free space in white, unknown in light grey with a U, extended pursuer and obstacle cells in grey with a G, and the robot in the centre).

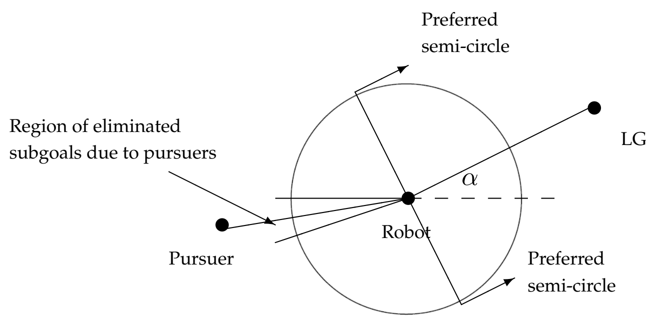


Figure 4.9: Illustration of the sampling of subgoals for bio-inspired proteanism.

The following procedure has been proposed to compute the proteanism subgoals, illustrated in Figure 4.9:

1. Drawing a circle of radius r_{sg} centred in the current robot location.
2. Calculating the distance between the robot and LG: $d(\text{robot}, LG)$, and the angle with the horizontal, α.
3. Computing subgoals as samples on the circle every $k\theta$ radians, with exact location in

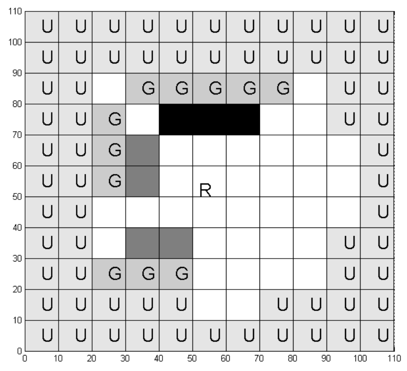
$$\begin{bmatrix} sg_x \\ sg_y \end{bmatrix} = \begin{bmatrix} r_{sg} \cdot \cos\theta \cdot \cos\alpha - r_{sg} \cdot \sin\theta \cdot \sin\alpha \\ r_{sg} \cdot \sin\theta \cdot \cos\alpha + r_{sg} \cdot \cos\theta \cdot \sin\alpha \end{bmatrix} \tag{4.14}$$

4. Eliminating the samples that fall in a probable pursuer or obstacle cell.
5. Computing the angles between the pursuers (centroid of the probable pursuer cells) and the robot, ω_p, for each one of the n_p pursuers. Eliminate the subgoals that approximate those angles.
6. Choosing a subgoal that is in the semi-circle closer to the local goal randomly, but if no subgoals are available, then choosing from the other semi circle (opposite direction to the local goal). This would mean that obstacles are occluding a direct route towards the local goal.

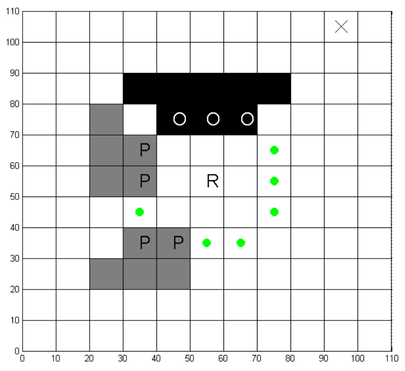
Figure 4.10 shows an example of the computation of subgoals with $\theta = \frac{\pi}{12}$ and $r_{sg} = 2$ cells, for a randomly generated environment with the same characteristics as the others in this chapter.

General proteanism

Guided proteanism previously presented is convenient when local or global navigation goals are defined, and they need to be reached as a specific purpose of the navigation task, as it is finding a hideaway. Nevertheless, if the only purpose of the navigation is avoiding capture by pursuers, a general proteanism like the one presented in Furuichi (2002) can be used.



(a) Extended, sensed environment



(b) $h^*(a,b)$ and available subgoals

Figure 4.10: Computed LG as $h^*(a,b)$ (probable obstacle and pursuer cells indicated with OC and the LG as X) and all subgoals, after eliminations in steps 4 and 5 of proposed procedure, as green circles, for an extended, sensed, randomly generated environment (obstacles in black, pursuers in grey, free space in white, unknown in light grey with a U, extended pursuer and obstacle cells in grey with a G, and the robot in the centre).

A more general implementation of proteanism *a priori* involves all remaining probable subgoals being considered in the random selection, so no priority is given to the ones closer to the local goal, as there is not a local goal. The object of the implementation of a more general proteanism is just to avoid capture when the pursuers are closer or when the robot is surrounded by pursuers, in a more reactive manner. If the pursuers are left behind, the robot could commute to keep navigating towards a local goal.

Subgoals as vectors

The attraction exerted by the subgoals to incite the proteanism clearly can be interpreted as a force, as in Furuichi (2002). The force has a unitary direction vector that leads towards the subgoal, with a magnitude given by the allowed displacement in the next navigation motion. Thus,

$$\vec{F}_{sg} = F_{sg} \cdot \vec{u} \tag{4.15}$$

where F_{sg} is the magnitude of the force (e.g., 1 cell), and $\hat{u}$ is the unitary direction vector pointing in the direction of the subgoal. $\vec{u}$ would be selected favouring a range of angles towards the local goal, and avoiding angles that could cause collisions with obstacles or leading directly towards pursuers. In general proteanism, as in Furuichi (2002), $\vec{u}$ would take any direction.

4.4.6 Updating the environment representation

The environment changes as the robot moves and senses different parts of the elements in the environment, and the pursuers update their locations following their own strategies. This means the need for updating the analysis of the environment, the local goal, the subgoals (at least the most immediate one), and then the potential functions or heuristics used to compute a new path accordingly.

If the environment is increasingly complex, and the robot can perform fast sensing and computation of a path following the analysis of the environment, resampling the sensing for shorter intervals of space and time would be convenient. This would produce a more continuous path that still avoids collisions, but changes the local goal and the future subgoals according to the new information of the environment. But those assumptions cannot be made for all the robotic physical platforms, as their sensing capabilities are different. If the environment is like the randomly generated ones with static obstacles, the updating could be performed in more separated intervals of space and time, for example after reaching a subgoal, or after reaching the local goal.

Another consideration for the updates is the velocity of the pursuers. If they are faster than the robot, more randomness in the navigation motions could increase the chances of avoid capture. Consequently, updates to modify the local goal, and the subgoals, could be performed more often than when pursuers are far away or comparatively slower.

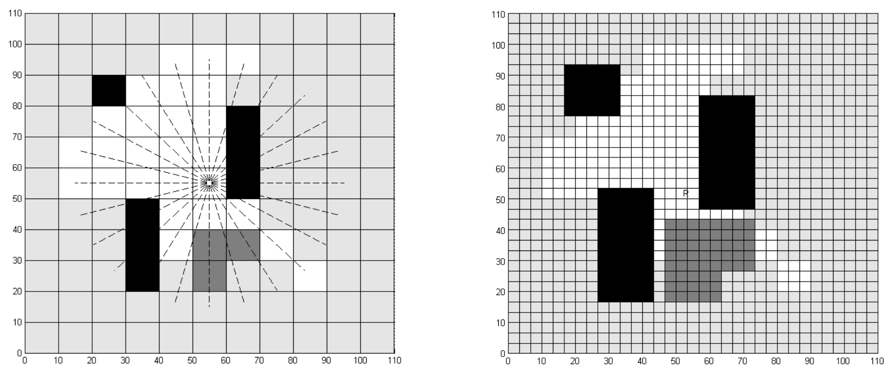
4.4.7 Modifying the resolution of the grid

The proposed 11×11 grid is the result of assuming a basic sparse sensing with an array of sensors that allows performing a sweeping of the surroundings of the robot as mentioned in Subsection 4.4.1. Such a discrete sample of the environment leads to a discrete environment representation, accordingly. Nevertheless, the resolution of the grid can be modified artificially to generate more samples, or by sensing using other kinds of devices that provide more samples and measures of the environment like a camera.

In the literature, many approaches for robotics path planning consider the sensing and acting of robots in continuous environments, or grids with high resolutions (e.g., 100×100 cells in Trevisan et al. (2006)). This involves developing continuous models of the environment where obstacles and other entities are represented by shapes and lines. Using more realistic sensing capabilities implies imperfect discrete information, particularly basic sensors like bumpers, infrared or ultrasonic. When using cameras, although the obtained image is the collection of discrete pixels, more information from the environment is collected and more continuous approximated environment representations can be computed.

As the designs presented in this project are intended for robots with different ranges of sensing capabilities, it has been important using different representations in terms of grid resolutions to compare the performance of the proposals. The analysis of the environment described in this chapter can be performed for the new high resolution grid, coming from artificially extending the environment, or from a higher resolution sensing (e.g., using a camera).

The original 11×11 grid can be extended artificially into a higher resolution grid by dividing the cells in smaller parts, with the corresponding probabilities for the different sensed elements. Adjacent cells around identified or probable obstacles and pursuers



(a) Extended, sensed elements (b) Increasing the resolution

Figure 4.11: Modification of the resolution of the grid with extension of obstacles and pursuers to avoid collisions, for an extended, sensed, randomly generated environment (obstacles in black, pursuers in dark grey, free space in white, unknown in light grey and the robot in the centre).

can become part of the obstacles and pursuers, so the robot is limited against moving its centroid towards those locations and avoiding collisions. Figure 4.11 shows the resizing of the grid into a higher resolution one, by dividing each original cell into 9 new cells.

The next section presents the computation of different kinds of potential function approaches, to obtain a function to compute a solution to the problem of navigation and escape from pursuers in the form of a path.

4.5 Potential functions for navigation and escape

The following subsections present the modification of different potential functions to use them for a final representation of the environment (including pursuers and *a priori* proteanism) to compute a solution to the problem of navigation and escape from pursuers in the form of a path: discrete potential functions, approximation of a boundary value problem (for the Laplace equation and for a perturbation equation), and wave-front potential functions. Subsection 4.5.4 presents a comparison of the resulting characteristics of the functions, according to the proposed modifications for the problem of navigation and escape from pursuers and a discrete representation from the analysis of the environment.

4.5.1 Discrete potential functions

A first approach to the solution of the problem of navigation and escape from pursuers was choosing popular functions in the discrete setting mentioned earlier in the chapter. Source or repulsion functions were used for the obstacles and the pursuers from the modified FIRAS function (Khatib, 1986) to incorporate the effect of equidistant goal and obstacles, i.e. when an obstacle and the goal are equidistant in a line from the location of the robot, which causes oscillations in the resulting path (Wang et al., 2002; Ding et al., 2005; Hui and Pratihar, 2008; Zhu et al., 2009; Lee et al., 2010; Tan et al., 2010; Takahashi et al., 2010; Pradhan et al., 2011). The sink or attraction function, for the local goal or the subgoal in turn when using protean navigation, was chosen to be the quadratic function also proposed by Khatib (1986).

The mentioned modified FIRAS repulsion functions for the obstacles and pursuers, limiting the influence of the repulsion force to the distance d_0, formulated for a 2-D grid with cell centroids in (a, b), have the form

$$U_r(a,b) = \begin{cases} \frac{1}{2}K_i \left(\frac{1}{d((a,b),i)} - \frac{1}{d_0}\right)^2 d((a,b),LG) & \text{if } d((a,b),i) < d_0 \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

where $d((a,b),i)$ is the Euclidean distance between a cell in the environment and a probable obstacle or pursuer cell i; $d((a,b),LG)$ is the Euclidean distance between a cell and the local goal; and K_i is a constant to regulate the magnitude of the repulsion.

The attraction function for the local goal or subgoals is defined as

$$U_a(a,b) = \frac{1}{2}K_{lg} \cdot d((a,b),LG)^2 \tag{4.17}$$

where K_{lg} is a constant to regulate the magnitude of the attraction by the local goal. When using subgoals, the distance is replaced by $d((a,b),SG)$, where SG is the corresponding active subgoal.

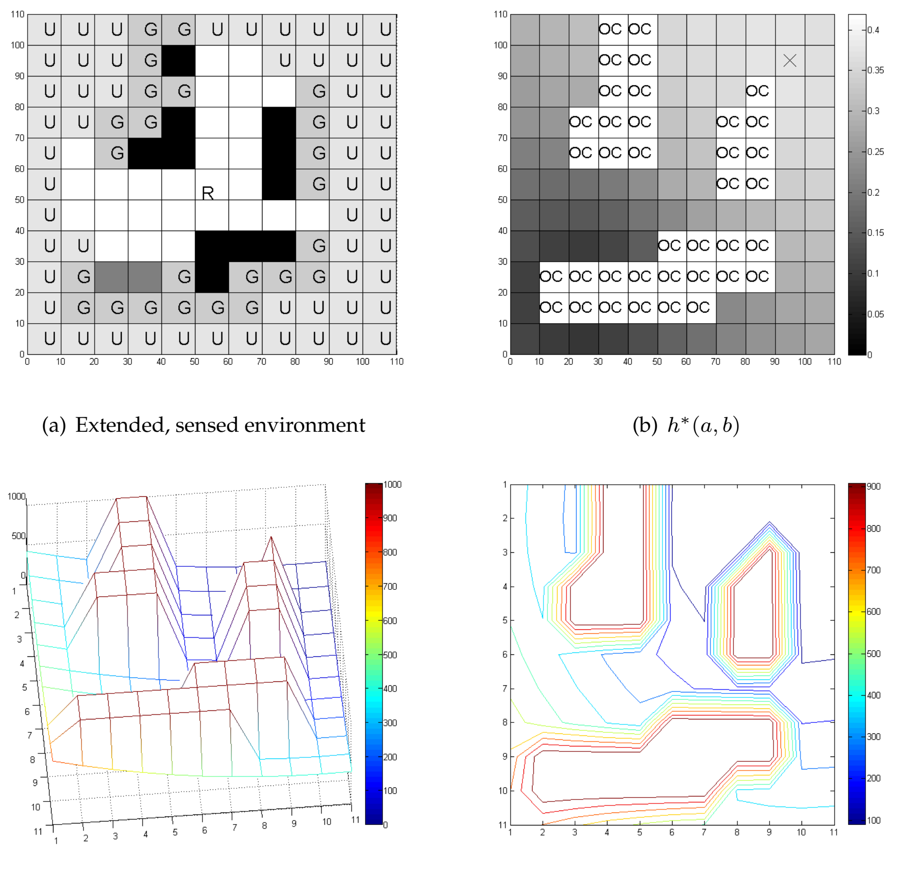
The total potential of each cell is given by

$$U_t(a,b) = \sum_{i_o=1}^{k_o} U_r(a,b) + \sum_{i_p=1}^{k_p} U_r(a,b) + U_a(a,b) \tag{4.18}$$

for k_o obstacle cells and k_p pursuer cells. The free cells do not produce any attraction or repulsion, but they are influenced by the other elements in the sensed environment.

Figure 4.12 shows an example of the resulting potential functions for the identified elements and the local goal from the best hideaways, in a randomly generated discrete environment of the same characteristics as the others in this chapter. The parameters used for the functions are: $K_{lg} = 0.1$, $K_i = 1$, $d_o = 1$ cell and cropping the maximum potential values to 1000 for visualisation.

The main disadvantage of using the potential functions presented in this section is the possibility of finding local minima when computing the path, despite the quadratic properties of the attraction function and the modification to the FIRAS function when obstacles and the goal are equidistant to the robot. Solutions to the problem of local minima, as mentioned in Subsection 3.3.6, include changing the use functions, or modifying the search method. Other approaches to compute a representation based on potential functions are presented in the following sections, starting with numerical solutions to harmonic functions from a boundary value problem that guarantee no local minima by design.



(a) Extended, sensed environment

(b) $h^*(a, b)$

(c) Total potential, $U_t(a, b)$

(d) Discretised gradient of $U_t(a, b)$

Figure 4.12: Computed LG as $h^*(a, b)$ (probable obstacle and pursuer cells indicated with OC and the LG as X), mesh plot of the total potential for each cell and contour plot of the discretised gradient, for an extended, sensed, randomly generated environment (obstacles in black, pursuers in grey, free space in white, unknown in light grey with a U, extended pursuer and obstacle cells in grey with a G, and the robot in the centre).

4.5.2 Approximation of a boundary value problem

An alternative potential function based final representation of the environment to compute a path is a discrete approximation of a BVP, solving a differential equation using numerical methods, boundary conditions, and given initial values. This alternative has been presented previously in Subsection 3.3.2. The Dirichlet initial boundary conditions are the most popular in the literature on path planning with potential functions (Connolly, 1994; Prestes et al., 2001; Alvarez et al., 2003; Vaščák, 2007; Pimienta et al., 2006). These conditions contemplate the following:

- The boundary of the goal is given a low potential value ($\phi_{goal} = 0$).
- The boundary of the obstacles and the limits of the known environment (walls) are given a high potential value ($\phi_o = 1$). If the robot is performing an exploration task, and the limits of the known environment are open, the boundary potential value is low ($\phi_{goal} = 0.1$) (Trevisan et al., 2006).
- The initial value of every other cell is given so that $\phi_{goal} > \phi > \phi_o$.

Other optional boundary conditions include the Newman, where the normal vectors to the obstacles and the limits of the environment (walls) are also given an initial condition of $\phi'_o = c$, where c is a constant (Shi et al., 2007).

Using harmonic equations (like Laplace) guarantees the absence of local minima mathematically. On the other hand, adequate boundary conditions need to be proposed to perform specific tasks like obstacle avoidance. Also, the many possibilities of iterative algorithms to compute the numerical approximations of the harmonic equations have different performances, in terms of the number of stencils of the approximation and the convergence to a solution with minimum error.

A first approach to the use of a BVP contemplates Dirichlet conditions to solve the Laplace equation as in Subsection 3.3.2. Subgoals are introduced and commuted for the local goal, to achieve the implementation of *a priori* proteanism. A second approach contemplates the implementation of *a priori* proteanism through a perturbation in the differential equation solved by the BVP with Dirichlet conditions. The perturbation has the purpose of guiding the resulting slope of potential values in the grid towards a specific direction (Silveira et al., 2010). Both approaches are presented in the following subsections.

Laplace equation and Dirichlet boundary conditions only

The first approach adds the pursuers as Dirichlet boundary conditions to solve the Laplace equation,

$$\nabla^2 \phi = 0 \tag{4.19}$$

The Taylor approximation of the Laplace equation, presented in Subsection 3.3.2, involves approximation of the continuous 2-D navigation plane of the robot into cells. The harmonic potential function described by the Laplace equation becomes a grid, and the potential values of each inner cell, (a, b), can be computed as the average of the adjacent cells (stencil),

$$\phi(a,b) = \frac{1}{4}[\phi(a+1,b) + \phi(a,b+1) + \phi(a-1,b) + \phi(a,b-1)] \tag{4.20}$$

The stencils are computed using iterative numerical methods, from boundary conditions for the different elements in the environment: the obstacles, the goal, the limits of the considered environment, and the pursuers. Pursuers can be incorporated in the same way as obstacles, with a high potential value like $\phi_p = 1$ for all the probable pursuer cells, and $\phi_o = 1$ for all probable obstacle cells. For the local goal or subgoal in turn that substitutes the local goal, a low potential value can be given, $\phi_{goal} = 0$. For all the other cells, any initial value $\phi_{goal} > \phi > \phi_o$ is given. The unknown limits of the grid need to have a boundary value $\phi_{goal} > \phi > \phi_o$, so that they are considered for the planning. But when using general proteanism, the boundaries of the grid can be used to attract the robot to unexplored regions to get away from pursuers, thus $\phi = 0$ (Prestes et al., 2001). The final potential values of the grid are computed using the Jacobi or SOR iterative methods, presented in Subsection 3.3.2.

An example of the resulting potential functions is shown in Figure 4.13, with the conditions mentioned above for the pursuers, obstacles and the local goal (no subgoals), and an initial value of any other cell of $\phi = 0.25$. The limits of the grid have a boundary value of $\phi = 0.1$. The potential values were calculated using Jacobi and SOR iterative methods, respectively.

The examples using the original 11×11 grid from the proposed discrete sensing illustrate the problem of iterative approximation methods when using low resolution grids. The computed potential discrete surfaces converge to the final result after 2 iterations for the Jacobi, and after 1 iteration for SOR. The obtention of an almost equipotential surface in the free space zone increases the difficulty of computing a path using only steepest descent, and another parameter like the distance to the goal needs to be considered in the path planning step.

Figure 4.14 shows the computation of the potential values from approximations to the Laplace equation using Jacobi and SOR iterative methods, for a higher resolution grid of 1/3 of the original size per side.

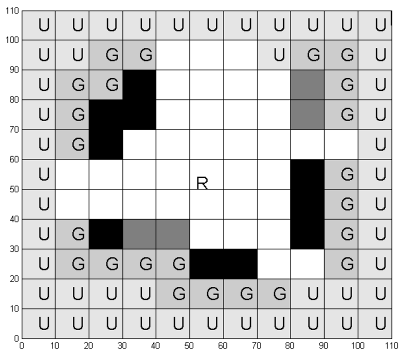
Dynamic perturbations for bio-inspired proteanism

A perturbation can be introduced to the potential functions to modify the slope of the discrete gradient (similar to Neumann initial conditions), by adding it to the Laplace equation,

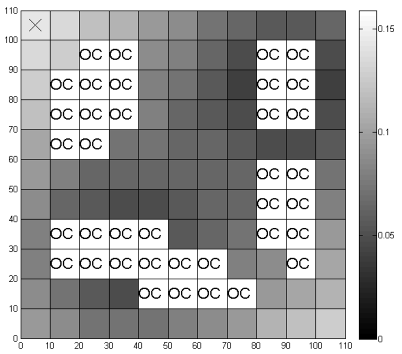
$$\nabla^2 \phi = \epsilon_p \vec{V} \cdot \nabla \phi \tag{4.21}$$

where ϵ_p is the strength of the perturbation, and $\vec{V} = [V_{x_1}, \ldots, V_{x_l}]$ is the perturbation in l dimensions (Trevisan et al., 2006). The magnitude of the perturbation is unitary, $|\vec{V}| = 1$.

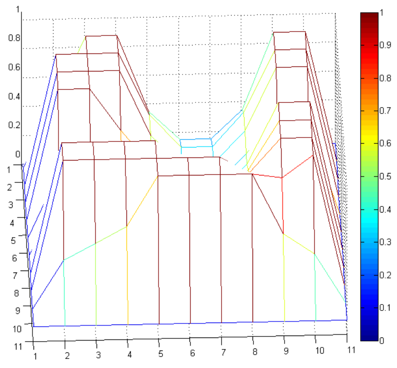
As in Subsection 3.3.2 for the Laplace equation, the Taylor series expansions about the point



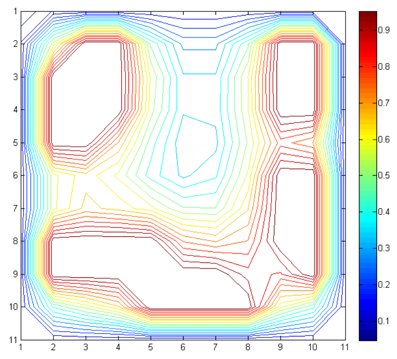
(a) Sensed environment



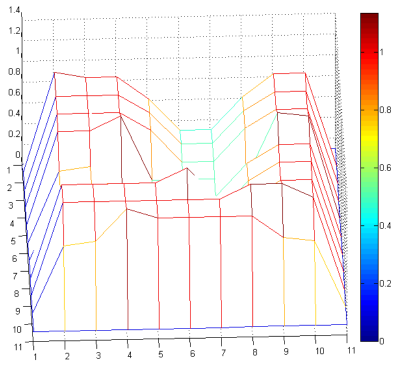
(b) $h^*(a,b)$



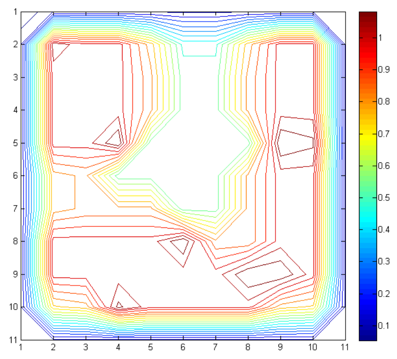
(c) $\phi(a,b)$ from Jacobi iterative method



(d) $\nabla\phi(a,b)$ from Jacobi iterative method

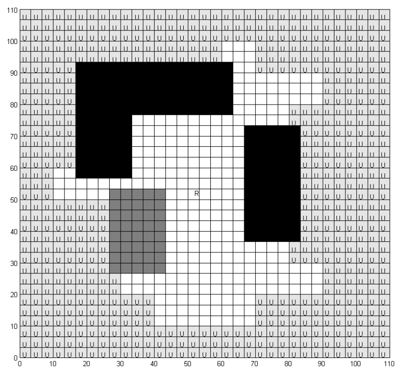


(e) $\phi(a,b)$ from SOR iterative method

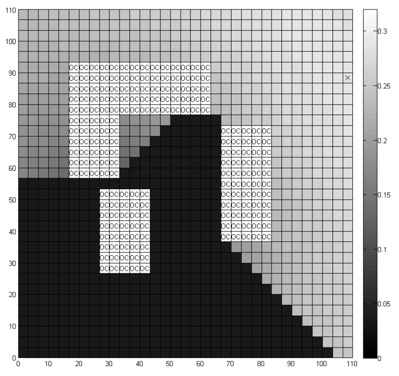


(f) $\nabla\phi(a,b)$ from SOR iterative method

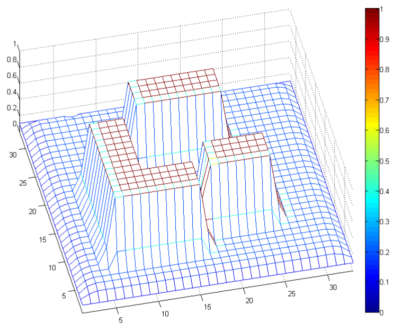
Figure 4.13: Computed LG as $h^*(a,b)$ (probable obstacle and pursuer cells indicated with OC and the LG as X), for a sensed, randomly generated environment (obstacles in black, pursuers in grey, free space in white, unknown in light grey with a U, extended pursuer and obstacle cells in grey with a G, and the robot in the centre). Mesh plot of the total potential for each cell and contour plot of the gradient using Jacobi and SOR iterative methods.



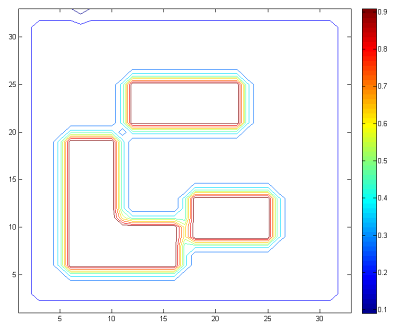
(a) High resolution sensed environment



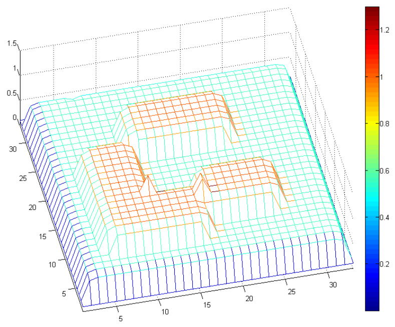
(b) $h^*(a, b)$



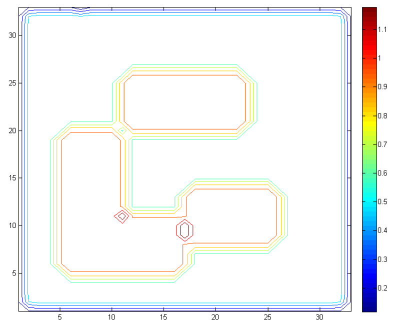
(c) $\phi(a, b)$ from Jacobi iterative method



(d) $\nabla\phi(a, b)$ from Jacobi iterative method



(e) $\phi(a, b)$ from SOR iterative method



(f) $\nabla\phi(a, b)$ from SOR iterative method

Figure 4.14: Computed LG as $h^*(a, b)$ (probable obstacle and pursuer cells indicated with OC and the LG as X), for a high resolution, sensed, randomly generated environment (obstacles in black, pursuers in grey, free space in white, unknown in light grey with a U, extended pursuer and obstacle cells in grey with a G, and the robot in the centre). Mesh plot of the total potential for each cell and contour plot of the gradient using Jacobi and SOR iterative methods.

x_1 for $\phi(x_1+\delta,\dots,x_l)$ and $\phi(x_1-\delta,\dots,x_l)$ results in

$$\phi(x_1+\delta,\dots,x_l) = \phi(x1,\dots,x_l) + \delta\frac{\partial\phi}{\partial x_1} + \frac{1}{2}\delta^2\frac{\partial^2\phi}{\partial x_1^2} + \frac{1}{6}\delta^3\frac{\partial^3\phi}{\partial x_1^3} + \mathcal{O}(\delta^4) \tag{4.22}$$

$$\phi(x_1-\delta,\dots,x_l) = \phi(x1,\dots,x_l) - \delta\frac{\partial\phi}{\partial x_1} + \frac{1}{2}\delta^2\frac{\partial^2\phi}{\partial x_1^2} - \frac{1}{6}\delta^3\frac{\partial^3\phi}{\partial x_1^3} + \mathcal{O}(\delta^4) \tag{4.23}$$

The two expansions are subtracted,

$$\phi(x_1+\delta,\dots,x_l) - \phi(x_1-\delta,\dots,x_l) = 2\delta^2\frac{\partial^2\phi}{\partial x_1^2} + \mathcal{O}(\delta^4) \tag{4.24}$$

and after some algebraic manipulation, considering $\mathcal{O}(\delta^4)\approx 0$,

$$\frac{\partial\phi}{\partial x_1} = \frac{\phi(x_1+\delta,\dots,x_l) - \phi(x_1-\delta,\dots,x_l)}{2\delta} \tag{4.25}$$

If the same procedure with the Taylor expansions is applied for every variable $x_1,\dots,x_l$, an approximation of the gradient, $\nabla\phi$, can be obtained in the form of

$$\begin{aligned}\sum_{l=1}^{L}\frac{\partial\phi}{\partial x_l} = \frac{1}{2\delta}\Big[&\phi(x_1+\delta,\dots,x_l) + \dots + \phi(x_1,\dots,x_l+\delta) - \phi(x_1-\delta,\dots,x_l) - \dots\\ &- \phi(x_1,\dots,x_l-\delta)\Big]\end{aligned} \tag{4.26}$$

Substituting the approximations of the Laplace equation (Equation 3.12 in Subsection 3.3.2) and the gradient (Equation 4.26) into Equation 4.21 results in

$$\begin{aligned}&\frac{1}{\delta^2}\Big[\phi(x_1+\delta,\dots,x_l) + \dots + \phi(x_1,\dots,x_l+\delta) + \phi(x_1-\delta,\dots,x_l) + \dots\\ &+ \phi(x_1,\dots,x_l-\delta) - 2l\phi(x_1,\dots,x_l)\Big] = \frac{\epsilon_p}{2\delta}\vec{V}\cdot\Big[\phi(x_1+\delta,\dots,x_l) + \dots\\ &\quad + \phi(x_1,\dots,x_l+\delta) - \phi(x_1-\delta,\dots,x_l) - \dots - \phi(x_1,\dots,x_l-\delta)\Big]\end{aligned} \tag{4.27}$$

from where, after some algebraic manipulation,

$$\begin{aligned}\phi(x_1,\dots,x_l) = \frac{1}{2l}\Big[\frac{1}{\delta^2}\Big[&\phi(x_1+\delta,\dots,x_l) + \dots + \phi(x_1,\dots,x_l+\delta) + \phi(x_1,\dots,x_l+\delta) + \dots\\ &+ \phi(x_1,\dots,x_l-\delta)\Big] - \frac{\epsilon_p}{2\delta}\cdot\vec{V}\Big[\phi(x_1+\delta,\dots,x_l) + \dots + \phi(x_1,\dots,x_l+\delta)\\ &- \phi(x_1-\delta,\dots,x_l) - \dots - \phi(x_1,\dots,x_l-\delta)\Big]\Big]\end{aligned} \tag{4.28}$$

For a 2-D environment defined as cells with location, (a,b), with a distance of 1 from the point or cell to the next (i.e. $\delta=1$), and a perturbation $\vec{V}=[V_a,V_b]$, the potential in a point is reduced to

$$\phi(a,b) = \frac{1}{4}\left[\phi(a+1,b) + \phi(a,b+1) + \phi(a-1,b) + \phi(a,b-1)\right] \\ - \frac{\epsilon_p}{8}\left[V_a \cdot [\phi(a+1,b) - \phi(a-1,b)] + V_b \cdot [\phi(a,b+1) - \phi(a,b-1)]\right] \tag{4.29}$$

Grouping terms, Equation 4.29 becomes

$$\phi(a,b) = \left(\frac{1}{4} - \frac{\epsilon_p V_a}{8}\right)\phi(a+1,b) + \left(\frac{1}{4} + \frac{\epsilon_p V_a}{8}\right)\phi(a-1,b) + \left(\frac{1}{4} - \frac{\epsilon_p V_b}{8}\right)\phi(a,b+1) \\ + \left(\frac{1}{4} + \frac{\epsilon_p V_b}{8}\right)\phi(a,b-1) \tag{4.30}$$

where the sum of all coefficients of the stencil equals 1 (Prestes et al., 2001).

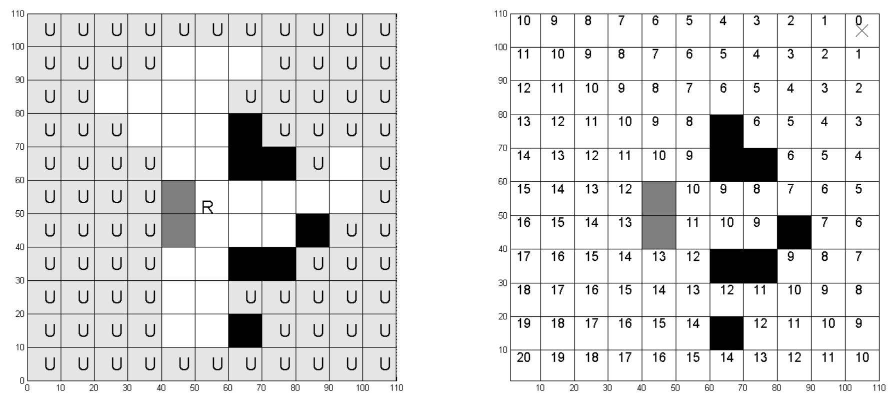
As in the previous subsection, the Dirichlet boundary conditions are the following: $\phi_p = 1$ for all the probable pursuer cells, $\phi_o = 1$ for all probable obstacle cells, $\phi_{goal} = 0$ for the local goal, any initial value $\phi_{goal} > \phi > \phi_o$ for all the other cells, and a boundary value of $\phi_{goal} > \phi > \phi_o$ for the limits of the grid, so that they are considered for the planning. When using general proteanism, the limits of the grid can have a boundary value of $\phi = 0$ to favour the exploration of new parts of the global environment as a possible navigation strategy away from pursuers (Prestes et al., 2001).

The discrete potential values for each cell produce surfaces (of the potential values) and contours (of the gradient), have an identical shape to the ones obtained without incorporating a perturbation in the Laplace differential equation (previous subsection) when the perturbation is directed towards the local goal or subgoal in turn. The next subsection presents another alternative method to compute potential functions that guarantee mathematically the existence of a global minimum in the potential values of the grid.

4.5.3 Wave-front potential functions

The wave-front method allows computing potential functions in a grid, avoiding the presence of local minima. Starting from all free space cells with a value of 0, a value is given to a cell, and then it is propagated with increments or decrements towards the remaining adjacent cells that still have a value of 0, until all the grid is covered (Choset et al., 2005). A path can be computed using steepest descent from any cell in the free space.

A wave-front potential function can be computed based on the local goal, LG, by assigning a value (e.g. 1) to the corresponding cell, and then propagating it. The local goal is computed as the best final location in the sensed part of the environment, according to functions of distance between the pursuers and the robot, and the probable hideaways behind obstacles. Propagating the local goal allows guiding the robot towards that location avoiding local minima problems. The resulting propagated potential values (from the local goal) are analogous to a cost function of the distance between any cell in the free space



(a) Extended, sensed environment (b) Wave-front potential function

Figure 4.15: Wave-front potential function with origin (minimum value) in the local goal (marked as X), showing the values of the free space cells (obstacles in black, and pursuers in grey), form an extended, sensed, randomly generated environment (obstacles in black, pursuers in grey, free space in white, unknown in light grey with a U, extended pursuers and obstacles cells in grey with a G, and the robot in the centre).

of the grid to the local goal. Figure 4.15 shows the computation of a wave-front potential function for a local goal.

Proteanism (*a priori*) can be implemented by substituting the local goal by the current active subgoal and recomputing the wave-front propagation. This mechanism also works for general proteanism, where no local goal is computed, but subgoals guide the robot away from pursuers in a protean manner.

An alternative form to implement the escape from the pursuers without computing the local goal, and without considering proteanism can be achieved by assigning the pursuer cells a high value (e.g., 100), and then computing their decreasing propagation. The total of all the different wave-front propagations for the pursuer cells can be computed as the maximum of the values for each cell,

$$\phi_{wf}(a,b) = \max_{\forall k_p} \phi_{k_p}(a,b) \tag{4.31}$$

where k_p are the pursuer cells, and $\phi_{k_p}(a,b)$ is their corresponding wave-front propagations. Then, steepest descent can be applied to guide the robot far from the pursuers.

4.5.4 Comparison of the used potential functions

Table 4.1 presents a summary of the main characteristics of the potential functions explained before, used in the designs for autonomous navigation and escape from pursuits. First impressions indicate the possibility of different kinds of local minima problems depending on the location of obstacles and pursuers in the grid.

Table 4.1: Comparison of the main characteristics of the used potential functions.

NAME	CHARACTERISTICS	EXPECTATIONS
Discrete potential functions	Modified FIRAS potential functions for obstacles and pursuers and a quadratic function for the local goal or subgoals, discretised to the centroids of the cells in a grid	Local minima problems in cells near obstacles; computationally cheap to calculate and update.
BVP for Laplace's equation with Dirichlet conditions	Use of Laplace's equation and relaxation numerical methods to compute a discrete approximation of the potential function, with a global minimum in the goal or subgoal, and other initial conditions for obstacles (highest values), edges of the grid, and any other cell.	A global minimum in the grid, but with the possibility of local minima problems due to the presence of flat surfaces (plateau) in the free space; complexity of the computation depends on size of the grid and convergence conditions.
BVP for Laplace and perturbation equation with Dirichlet conditions	Use of Laplace's equation with an added perturbation, and relaxation numerical methods to compute a discrete approximation of the potential function, with a global minimum in the goal or subgoal, and other initial conditions for obstacles (highest values), edges of the grid, and any other cell.	A global minimum in the grid, but with the possibility of local minima problems due to the presence of flat surfaces (plateau) in the free space; similar results to the Laplace equation without perturbation.
Wave-front potential functions	A wave-front expansion is used from the local goal or subgoal (global minimum) towards the other cells in the grid, then assigning higher values to obstacle cells.	A global minimum in the grid, but with the possibility of local minima problems near obstacle cells depending on their location; computationally cheap.

4.6 Planning the path

The search method that computes the path from the scalar potential functions that represent the environment is the last step in the computations towards the solution to the problem of navigation and escape from pursuers. The flow diagram in Figure 4.16 shows an update of Figure 4.2, incorporating the different decisions that need to be made before planning the high level path according to the analysis of the environment, where the dashed lines indicate the optional feature of computing a higher resolution representation of the sensed environment. These decisions involve: *(a)* the commutation from normal navigation avoiding obstacles to escape from pursuers when they have been detected in the environment; *(b)* the addition of proteanism through subgoals or basing the navigation only on the computed local goal in terms of distances and possible hideaways; and *(c)* the use of general proteanism instead of computing a local goal and combining it with guided proteanism. The process of analysis and planning is updated according to new information from the environment.

The most common form in the literature to compute a path when using potential functions to represent the environment is following the gradient descent, or steepest descent for a discrete approach as inKoren and Borenstein (1991). Other search methods include depth-first based on a sphere (Al-Sultan and Aliyu, 1996), subgoals (Bell, 2005), and optimisation of an original random path (Warren, 1989). Subgoals are already implemented when using proteanism, in both guided or general form, and connections between the commuted subgoals need to be done also by a search method. A modification of steepest descent has been considered as the search method behind the connection of a location and the local goal, or the different subgoals commuted in the proteanism.

The following paragraphs present a simple model of the robot, where a path can be computed as a set of cells (e.g., coordinates of the centroids) from a start to an end in the locally sensed part of the environment. The steepest descent algorithm for potential functions is also presented, adding some examples using a local goal and guided proteanism (subgoals). Then, the modification to minimise distances for equal potential values is presented.

Simple model of the robot Considering a very simple model of the robot, as a cell in the grid, its position is given by the triple $[x, y, \theta_r]$, where (x, y) is the location measured with respect to a general frame of reference (e.g., distances from an origin). θ is the direction of the robot, and when planning for coordinates in the plane, θ is assumed to point in the direction of the next displacement. The robot moves with a maximum velocity of v_{max}, thus the maximum displacement that it can perform in a straight line, in a unit of time with that velocity, is

$$x_{max} = v_{max} \cdot t \tag{4.32}$$

where t is the interval of time. The motion of the robot can be considered deterministic for environment purposes. In real life, the motions of the robot are not limited to straight lines joining cells in a grid, as the steering constraints need to be considered, and the result can be stochastic (i.e., without certainty that the robot will move exactly to the locations

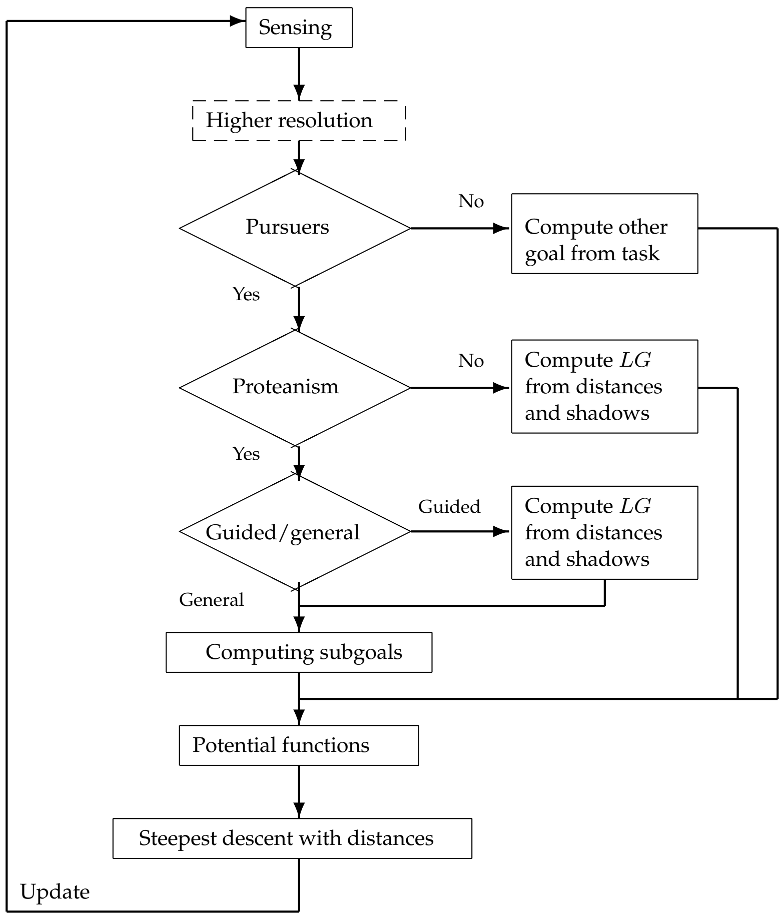

Sensing
Higher resolution
Pursuers
No
Compute other goal from task
Yes
Proteanism
No
Compute LG from distances and shadows
Yes
Guided/general
Guided
Compute LG from distances and shadows
General
Computing subgoals
Potential functions
Steepest descent with distances
Update

Figure 4.16: Flow diagram of all the steps and decisions in the analysis of the environment and path planning according to the presence of pursuers, the application of proteanism, and the kind of proteanism (guided or general). The dashed line represents the option of computing a higher resolution representation of the sensed environment. Through the selection of proteanism or not, a subgoal based (protean path) and a steepest descent only can be computed.

given by the path). A path given by sets of coordinates that represent cells in a grid can be tracked according to the real capabilities of the robot, serving as a general guide to how the general motion should happen, but not specific to all instants of time, as the tracking can be performed by a low level control.

Steepest descent The gradient descent of a continuous function is used to find a global optimum (minimum) by an iterative algorithm. Starting from a configuration in the space, v_k, every iteration computes a new configuration that is closer to the optimum,

$$v_{j+1} \leftarrow v_j - \alpha_{GD} f'(v_j) \tag{4.33}$$

where $f'(v_j) = \nabla f(v_j)$ is the gradient of the function $f(v_j)$, and α_{GD} is a variable step size for the approximation towards the optimum (Johnson and Picton, 1994). In discrete search spaces, the gradient is approximated by

$$\nabla f \approx f(v_{j+1}) - f(v_j) \tag{4.34}$$

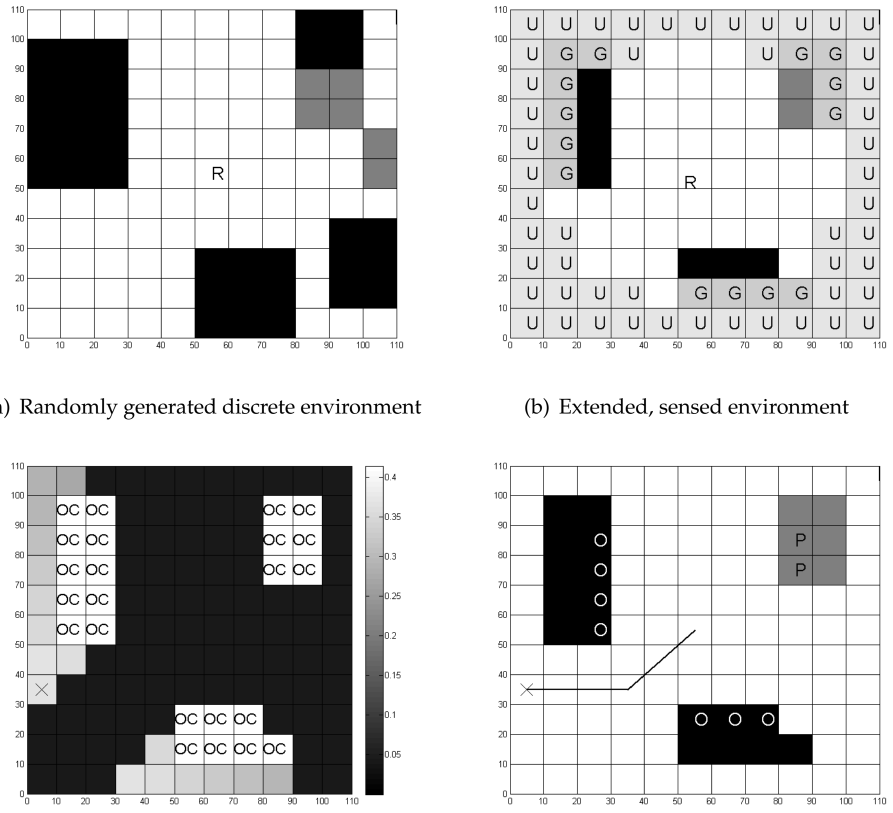
where v_j is a configuration of a discrete set (Johnson and Picton, 1994). The step size is fixed to the distance between the discrete configurations in the space. Every iteration, the neighbour configurations, $V = \{v_{1,j}, ..., v_{u,j}\}$, are examined to determine which one leads one step closer to the global optimum,

$$v_{j+1} \leftarrow \min_{\forall v_{u,j} \in V} f(v_{u,j}) \tag{4.35}$$

Figure 4.17 shows an example of a path towards a local goal computed by steepest descent, without subgoals, using the discrete potential functions mentioned in Section 4.5.1 with parameters $K_g = 0.1$, $K_i = 1$, $d_o = 1$ cell and a maximum value of 1000. The approximated discrete environment contains a maximum of 5 square randomly placed obstacles with a maximum size of 9 cells each, and a maximum of 2 square randomly placed pursuers with a maximum size of 4 cells each. They have been sensed with a radius of 3 cells, and extended with a probability of 0.5 as previous examples.

Figure 4.18 shows an example of the calculation of a path that connects subgoals using steepest descent. The subgoals were generated for $r_{sg} = 2$ cells and a sampling angle of $\theta = \frac{\pi}{12}$, up to a maximum of 10 subgoals. The steepest descent from a subgoal to the other was computed from the discrete potential functions in Section 4.5.1 with parameters $K_g = 0.1$, $K_i = 1$, $d_o = 1$ cell and a maximum value of 1000. The environment has the same characteristics as the example in Figure 4.17.

Adding the distance minimisation Steepest descent minimises the potential function values within the adjacent cells, but when those values are equal, another decision criterion can provide a better option than selecting one of the equal value cells randomly, which might lead directly towards a pursuer. Thus, minimising the distance towards the local goal, or active subgoal has been added to the steepest descent algorithm.



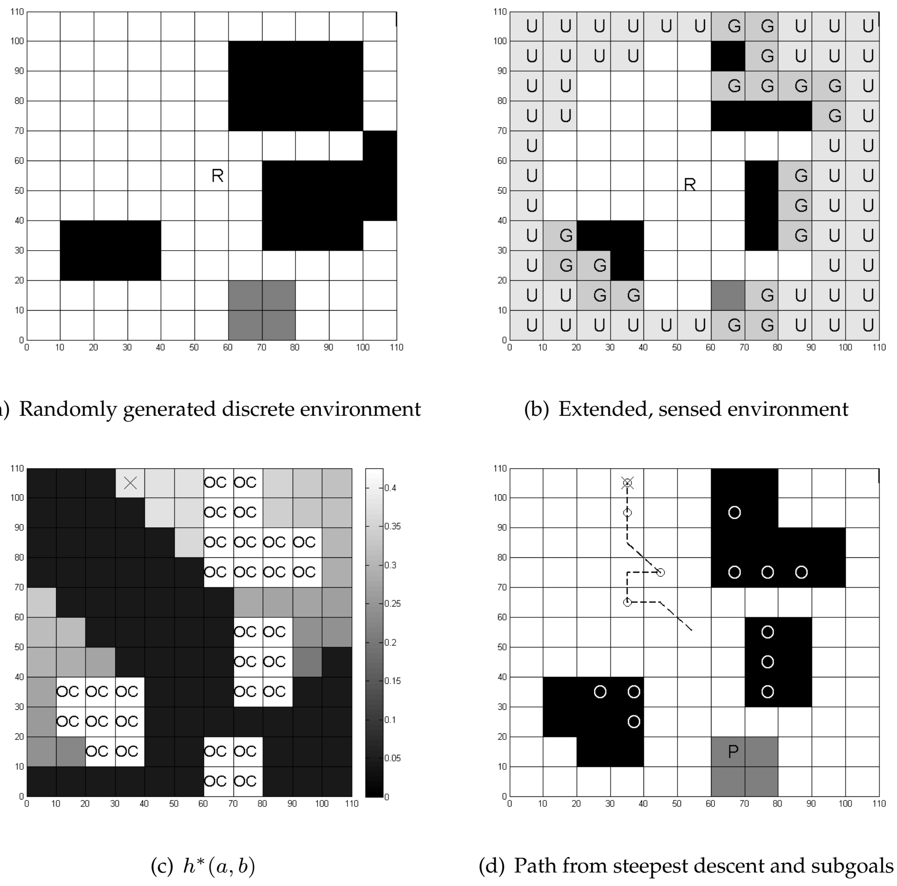
(a) Randomly generated discrete environment

(b) Extended, sensed environment

(c) $h^*(a,b)$

(d) Path from steepest descent

Figure 4.17: Computed LG as $h^*(a,b)$ (probable obstacles and pursuers cells indicated with OC and the LG as X) and resulting path using steepest descent (sensed obstacles in black with an O, sensed pursuers in grey with a P, extended probable obstacles in black, extended probable pursuers in grey, and the LG as X), for a randomly generated discrete environment, extended and sensed (obstacles in black, pursuers in grey, free space in white, unknown in light grey with a U, extended pursuers and obstacles cells in grey with a G, and the robot in the centre).



(a) Randomly generated discrete environment

(b) Extended, sensed environment

(c) $h^*(a,b)$

(d) Path from steepest descent and subgoals

Figure 4.18: Computed LG as $h^*(a,b)$ (probable obstacles and pursuers cells indicated with OC and the LG as X) and resulting path using steepest descent and subgoals (sensed obstacles in black with an O, sensed pursuers in grey with a P, extended probable obstacles in black, extended probable pursuers in grey, the LG as X and subgoals as circles), for a randomly generated discrete environment, extended and sensed (obstacles in black, pursuers in grey, free space in white, unknown in light grey with a U, extended pursuers and obstacles cells in grey with a G, and the robot in the centre).

Considering Equation 4.35, the next navigation action is computed as the action (motion towards an adjacent cell) that minimises the potential within the surrounding cells. If moving to more than one surrounding cells minimises the potential (different actions), the cell that minimises the distance to the local goal (while also minimising the potential) is chosen instead,

$$v_{j+1} = \begin{cases} \min \phi(a,b) & \text{if } MP = \{\text{one action}\} \\ \min d(MP, LG) & \text{if } MP = \{\text{different actions}\} \end{cases} \tag{4.36}$$

where MP is the subset of adjacent cells that minimise the potential of the current robot location in the grid, with size 1 if only one action achieves the minimisation, or larger than one if different actions minimise the potential. $d(LG, MP)$ is the distance between the local goal (or correspondingly an active subgoal being used for guided or general proteanism) and the centroid of a cell in the subset MP.

***A posteriori* proteanism** The paths generated by search methods like steepest descent are optimal and predictable if the locations of the obstacles, the pursuers, the goal and the robot are known or can be computed. Pursuers with embedded intelligence that allows them to predict probable optimal trajectories of a robot are a real challenge to avoid. Thus, generating a path with some degree of randomness that deviates from the predictable optimum but still takes the robot far from pursuers is needed.

The implementation of subgoals to obtain a deviation from the original steepest descent obtained path could not be enough when the pursuers are closer. *a posteriori* enhancement has been proposed in this thesis to increase the complexity of the movements whilst doing the planned navigation continuous motion. An easy form to understand and implement the proteanism is through adding noise to a whole path, or sections of it, to avoid long, constantly linear motions. Dynamic parameters (velocity, acceleration, steering) can be considered when designing the noise. Unfortunately, the new path is not optimal in terms of distance or time towards a navigation goal, and the consumed energy to perform a protean motion can be more than linear motions towards the goal.

4.7 Discussion

The definition of the problem of navigation and escape stated for the work presented in this thesis is based on general concepts of 2-D navigation. There are similarities between this problem and the problems of stealth navigation, and the problem of simple navigation, but the presence of a dynamic threat provides a need for a response that motivates the designs presented in the chapter. Also, the problem of escape in 3-D (particular to aircraft) involves different kinds of possible strategies to achieve a solution, as changing the altitude can be used to avoid obstacles, and even avoid capture or interception by the pursuers.

The problem can become more complex when adding localisation and mapping, as the robot needs to store additional information and use it to make decisions. Knowing the environment from exploration and mapping brings advantages of more information and certainty, that can be used to generate larger and more complete paths. Nevertheless, any robot that is deployed in a random place without knowing it in advance might encounter threats and its integrity depends on its capability to respond and try to preserve itself. Even when the robot can learn to respond to the environment, some innate capabilities against immediate threats can be the difference between being destroyed and succeeding in its task.

Considering the ideas of self-preservation, as part of designing mobile robots for any kind of task, a general high level approach allows universality in hardware platforms. That is, the analysis of the environment can be adapted to different sensing capabilities, and the path can be adapted to different kinds of hardware, intrinsically in the low level control to follow a path. The size of the grid can be rescaled according to the resolution of the sensing (e.g., using a camera, using a laser sensor that sweeps around), and the certainty of the information can be reproduced as probabilities. Nevertheless, the sensing has to be able to detect pursuers coming from any direction, just as humans and animals can turn around to see their surrounding. The capabilities of the sensing dictate the updates in the analysis of the environment and the computed path. The ability of sensing and the possibility of recognition of the different elements of the environment are basic requirements for the success in the navigation and escape.

In this thesis, the performance of the algorithms has been demonstrated through simulations, as implementation in real robots along with low level controllers represents another major challenge. The discrete path that is obtained can be connected through continuous motion according to real dynamic constraints. Having more information from the environment (through finer sensing) leads to higher resolution grids with more identified elements, and produces a path that approximates to higher continuity and accuracy. On the other hand, more sensors and better quality of information comes at a higher financial cost, as besides the sensing, accommodating the devices could require modifications to conventional robotic platforms available in the market.

Following the same line of adaptability to different hardware capabilities, a variety of potential functions have been considered for the path planning. They have been adapted to the problem of navigation and escape from pursuers, to provide a wide range of options to compute potential values according to the desired performance and any other additional requirements of a future task (like exploration, or area coverage). The different potential functions have been evaluated to compare their performances under similar constraints in simulations. Other alternative potential functions, like vector potential fields are useful for the final representation of the environment and path planning, as instead of using steepest descent, following the forces leads to the formulation of a navigation path. Nevertheless, investigating those more complex potential functions is a matter for future research.

Another aspect that has been proposed to solve the problem of navigation and escape, in a way to promote its adaptability, is the computation of subgoals. Proteanism has been implemented as both guided (e.g. when animals perform zig-zag motions, but also move towards their nest) and general (moving anywhere possible without any other temporal purpose but to avoid capture), to complement any other task requirement like a global objective of navigation, or simple exploration. The chosen implementation of the proteanism does not intend to mimic the real behaviour of any animal species for now, but it generalises the biological concept in a more basic context for its application in 2-D navigation over a robotic platform (generally more limited in motion capabilities than an animal of any kind). The two implementations can be used in conjunction with any of the potential functions.

Finally, the implementation of *a posteriori* proteanism has been proposed tentatively as an alternative to provide a measure against pursuers, by enhancing any computed path through random motions. Using *a posteriori* proteanism does not depend on the proposed analysis of the environment and consequent computation of a path through steepest descent. This alternative does not necessarily lead away from pursuers, but only prevents the predictability of linear motion.

4.8 Concluding remarks

The chapter started by presenting the particularities of the problem of navigation and escape from pursuers, including the different elements that form it, and its differences to other widely studied navigation problems. Then, the main ideas behind the proposed approach to the solution of the problem are presented:

- A deliberative control approach through a set of high level path planning algorithms.
- The different considerations about the elements of the environment that have been used to delimit the proposed designs, principally the use of sensors in real life, and their limitations of providing information about the environment.
- The use of different solutions from a path planning and game theory perspectives, the latter combined with reinforcement learning, and both in conjunction with bio-inspired features from feasible individual evasion mechanisms.
- The methodology for designing path planning algorithms, by dividing the process in the analysis of the environment and the proper planning stage where the actions in the plan are chosen.
- The use of different potential functions that have different properties regarding the presence of local minima, the computational easiness and the representation of the elements in the environment.

The two different bio-inspired features incorporated in the designs are presented: the use of refuges as a temporary navigation solution against capture, and the use of proteanism (random motions) in the navigation to confuse intelligent pursuers and delay the capture as much as possible. The implementation of both are based on general behavioural models adaptable to robotics, but not particular to any animal species.

Subsequently, the chapter presented a proposal for the analysis of the environment from sensed local elements, identifying pursuers, obstacles and free space in a discrete grid. The proposed analysis also provides information about the distances and possible hideaways, locating the best possible location according to the sensed environment (the local goal). A final representation is computed in the form of potential functions, for which the following have been considered and adapted for the computation of paths to solve the problem of navigation and escape from pursuers:

- Discrete potential functions, based on the modified FIRAS function and a quadratic goal.
- Solutions to the Laplace differential equation using a BVP and Dirichlet boundary conditions, through an approximation by numerical methods (Jacobi and SOR iterative methods).
- Solutions to the for the Laplace with perturbation differential equation through a boundary value problem, also with Dirichlet boundary conditions and an approximation by numerical methods.
- Wave-front potential functions, where the potentials become a function of the distance to the navigation objective.

A modified version of the steepest descent search method is presented, where the distances to the navigation objective (local goal or subgoal) are considered when more than a possible following motion action has the same potential value. This modification allows simplifying the computation of paths that lead to the navigation objective following the discrete potential functions, and also minimising the distances.

Finally, the discussion of the chapter highlights certain aspects of the proposed designs and adaptations of the potential functions to the problem of navigation and escape. Some of the most important aspects include the extension of the analysis of the environment to different sensing capabilities, and to other more broad navigation objectives according to the task that the robot needs to perform. The main novel aspects of the designs presented in this thesis have been introduced in this chapter: using bio-inspired behaviours to solve a navigation problem, in the form of hideaways to guide the navigation and randomly selected subgoals to produce protean motions. The main advantages of the designs in this chapter are:

- The discrete and local representation of the environment allows considering possible limited knowledge due to the sensing capabilities.
- The proposals allow the extension of the grid to any shape and resolution, according to the sensing capabilities of the robot, and the use of adequate potential functions from the proposed options (adaptability).
- The pursuers are considered as part of the environment.
- The bio-inspired aspects of hideaways and proteanism allow constructing anti-predator behaviours from path planning techniques, to offer alternative solutions to the previously proposed based on complete knowledge of the adversary.
- The chosen representation of the environment and the path planning based on potential functions are computationally cheaper than other combinations.

The next chapter presents different series of simulations, to evaluate the proposed general approach for the environment analysis based on local sensing, and the path planning using potential functions, under different parameters. Combinations of updates in the environment, changes in the resolution of the grid, computation of the proteanism, and the presence of elements are some of the main parameters variated in the simulations. Also, the different potential functions described in this chapter as options for the computation of the path are evaluated.

Chapter 5

Simulations with bio-inspired paths and pursuers

The previous chapter presented the proposed designs to solve the problem of navigation and escape from pursuers, for robots that navigate in a 2D plane (like ground vehicles), and therefore need preventing collisions with fixed obstacles whilst avoiding capture. The analysis of the environment, resulting in potential functions to produce a navigation strategy incorporating bio-inspired features and proteanism, is evaluated to analyse the performance of different variations. The variations include incorporating proteanism in the planning of the motion or not, different updating times of both the analysis of the environment and planning and changes in the resolution of the grid.

The main features to evaluate are:

1. Success of the proposed environment analysis (from sensing to computing the potential functions), by analysing the computation of paths that reach a chosen objective.
2. The different proposed variations within the environment analysis (potential functions).
3. The proposed *a priori* proteanism implementation, compared to traditional steepest descent paths towards a chosen objective, in the increase of the survival time of the robot.
4. Navigation towards a dynamic reasoned objective according to the demands of the task: collision avoidance and escape from pursuers, compared against random navigation (general proteanism).

The used parameters for all simulations in this chapter are mentioned in Section 5.1, including the main features of the environment, and any other constant used for the sensing, *a priori* proteanism, and potential functions. The section also describes the programming of the pursuers (followers).

Different groups of simulations were designed and performed to evaluate the proposed path planning schemes for navigation and escape from pursuers in a 2-D plane. The groups of simulations are presented in sections throughout the chapter:

1. Simulations with static environments (by not updating the sensing) to evaluate the computation of paths that lead to a specified location, using the different kinds of potential functions, the proposed environment analysis, and *a priori* proteanism through subgoals (Section 5.2).
2. Simulations to evaluate the proposed environment analysis and path planning when simulating the update in the sensing (as it would happen in a real life task) while the robot navigates in the environment and detects different elements. The evaluated features include the performance of the different kinds of potential functions, and the performance when using *a priori* proteanism. Pursuers are also simulated, with three different velocities in terms of the velocity of the robot (Section 5.3).
3. Simulations with static environments (as in Section 5.2) and a higher resolution grid (by dividing the original grid from sensing), to compare the performance of the algorithms when the resolution is changed and the path is approximated to a more continuous one (Section 5.4).
4. Simulations to evaluate the performance of the proposals (as in Section 5.3) but using a higher resolution grid (Section 5.5).
5. Simulations to compare the success of a general proteanism (random navigation) approach against the use of a guided proteanism approach (towards the local goal), simulating the motion of the pursuers for different velocities (Section 5.6).

Each section contains flow diagrams of the computations, along with visual examples of the resulting paths for the randomly generated environments, a statistical analysis of the results on the success of the paths, and comments about the results. A more general discussion of all the results is presented in Section 5.7, followed by the conclusions of the chapter in Section 5.8.

5.1 Parameters for the simulations

The previous chapter shows several examples of the computed resulting features of the environment analysis, including the local goals from functions in terms of the shadows and the distances, and finally the representation in terms of potential values. The same procedures were used to compute the results shown in the simulations contained in this chapter.

The simulations have been divided into two main areas according to the kind of proteanism:

1. Guided proteanism, using a local goal, LG, to guide the robot towards a better location according to its circumstances. These simulations have been divided according to the resolution of the used grid, into:

Table 5.1: Variation of parameters for the validating simulations.

FEATURE	VARIATIONS
Environment	Different environments with obstacles, randomly distributed.
Sensing	Different extensions of the sensing by adding probable elements or just based on the information from sensing.
Grid resolution	Using different resolutions of the grid from sensed and probable elements.
Computation of paths	Paths with and without proteanism in the presence of pursuers, and the modified gradient descent search method.
Presence of pursuers	Different numbers of pursuers in the environment, randomly located. Pursuers with different dynamic capabilities in terms of their velocity of navigation.
Proteanism	Calculation of proteanism in different forms: long reaching or short reaching by varying the radius r_{sg}.
Updating computations	Updates of the environment after different intervals of time, measured as navigation steps.

(a) Low resolution grid

(b) High resolution grid

Both sets of simulations with different grid resolutions include the variation of the update of the analysis of the environment. Simulations with no updates of the environment analysis after the initial sensing were performed to evaluate the reachability of a local goal, LG. Then, simulations with updates of the sensing and planning every certain number of navigation steps according to the grid resolution were performed to analyse a more realistic scenario. Different numbers of pursuers randomly placed were used in the simulations.

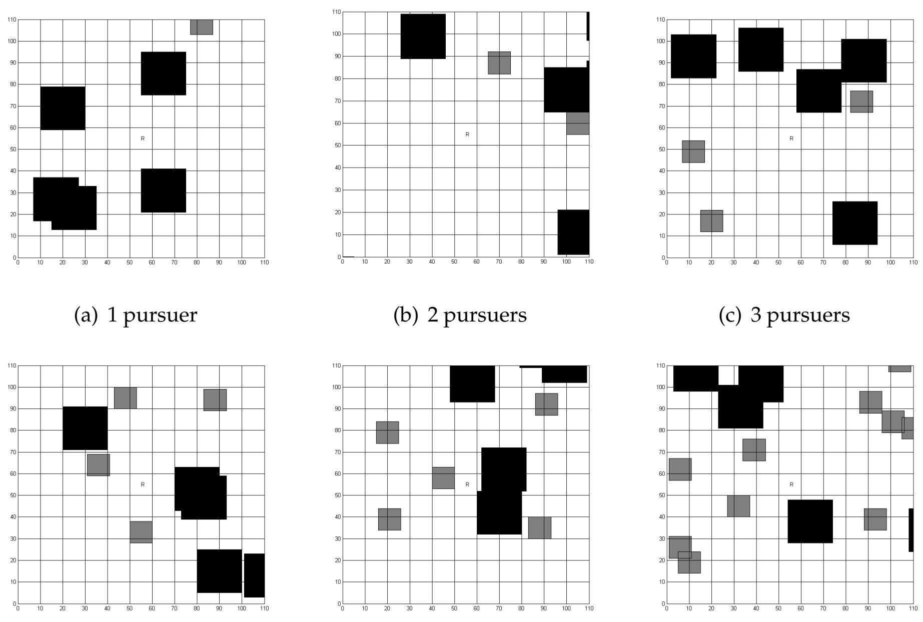
2. General proteanism, where the only objective of navigation is the prolongation of the time before capture by random motions away from pursuers.

All the sets of simulations included variations of the computation of subgoals, by increasing their radius of placement according to the resolution of the grid. Varying the value of the radius alters the length of duration of each subgoal in terms of distances. The tradeoff between planning in advance for many navigation steps and using unpredictability in the motion due to the randomness of switching subgoals constantly, needs to be considered to define an adequate value for the radius. Table 5.1 summarises the variations of the main features in the simulations, as explained.

The simulations were run for all the potential functions of interest: popular potential functions (modified FIRAS and quadratic goal), solutions to differential equations (harmonic and with perturbation) by a BVP (approximated numerically by Jacobi and SOR iterative methods) and potential functions from a wave-front expansion, in an environment represented as a grid. The results were compared to determine the performance of the different potential functions under variations in the proposed sensing and planning approach. Table 5.2 presents a summary of all the values used for the varied parameters in the simulations, including the different potential functions.

Table 5.2: Values used for each variable in the simulations.

VARIABLE	VALUES
Iterations	1000 (randomly generated environments) for simulations without updating the sensing for low resolution grids, 100 for the high resolution case. 100 for simulations updating the sensing and the pursuers for the low resolution, and 50 for the high resolution case.
Low resolution grid	Grid of 11×11 cells
High resolution grid	Grid of 33×33 cells
Obstacles	Five square obstacles, four times larger than the dimension of the robot, randomly placed in the environment.
Sensing	Sensing performed with a maximum radius of $3 \cdot$ size robot and an incremental angle of $\frac{\pi}{12}$ radians.
Extension of the sensing	For extended sensing, a probability of 0.5 was used for the adjacent cells behind sensed obstacles and pursuers. For unextended sensing, the probability of those adjacent cells is 0.
Computation of *a priori* proteanism	The following values for the radius (and consequently commutation) were used: r_{sg} = size cell $\times \{2, 3, 4, 5, 6\}$ for the low resolution grid, and r_{sg} = new size cell $\times \{3, 6, 9, 12, 15, 18\}$ for a higher resolution grid. Samples on the circle were taken each $\pi/12$ radians. A maximum of 10 subgoals were allowed for the low resolution grid, and a maximum of $10 \times div = 30$ subgoals were allowed for higher resolution grids obtained from dividing the original grid in $div = 3$ segments.
Pursuers	Pursuers were considered of the same size as the robot. The number of pursuers, randomly placed in the environment at the start of each experiment are: $1, 2, 3, 4, 5, 10$. Different velocities were also considered for the pursuers, measured from the velocity of the robot: $v_p = v_r, 2v_r, \frac{1}{2}v_r$.
Updating computations	Simulations without updates of the first sensing were used to assess if the robot is able to compute a path that leads to a local goal in the case of the guided proteanism. Simulations with updates were used to assess the success of the escape, for every 1,2,3,4,5,6 navigation steps for low resolution grids, and 3,6,9,12,15,18 navigation steps for high resolution grids (same distances as in low resolution grids).
Potential functions	$K_g = 0.1$, $K_i = 1$, $d_o = 1$cell, and a maximum potential value of 1000 in a cell.
Approximated solution to a BVP for the Laplace equation	Dirichlet boundary conditions: $\phi_{o,p} = 1$ for all the probable pursuer and obstacle cells; $\phi_{goal} = 0$ for the local goal; $\phi = 0.2$ for the limits of the grid with guided proteanism. Initial value of $\phi = 0.2$ for any other cell.
Approximated solution to a BVP for the Laplace equation with perturbation	Same boundary conditions. $\epsilon_p = 0.1$, perturbation as a unitary vector towards the local goal when using guided proteanism.
Wave-front potential functions	Starting value from local goal of 0, and increasing in steps of 1 unit.



(a) 1 pursuer (b) 2 pursuers (c) 3 pursuers

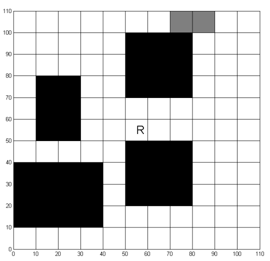
(d) 4 pursuers (e) 5 pursuers (f) 10 pursuers

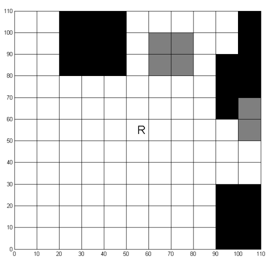
Figure 5.1: Examples of randomly generated environments with obstacles and different amounts of pursuers, for the simulations.

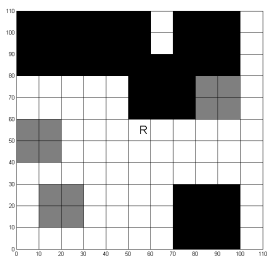
Generated environments

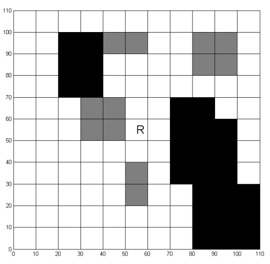
Square scenarios with 5 obstacles randomly placed were used for the simulations in this chapter, containing 1, 2, 3, 4, 5, and 10 pursuers. The original size of the generated obstacles is of 4 times the size of the robot (of 1 cell of the original sensing grid, 11×11 cells). The original size of the generated pursuers is the same as the robot (1 cell). Examples of these generated environments are shown in Figure 5.1, for all the numbers of pursuers. Although the location of the elements is random originally, the elements are approximated to an occupancy grid as part of the sensing simulation, as shown in Figure 5.2 for the same environments as Figure 5.1. The simulation structures generated randomly distributed environments, according to the specified number of pursuers, to compute and compare different paths. These computer generated environments were used in all the simulations presented in the chapter.

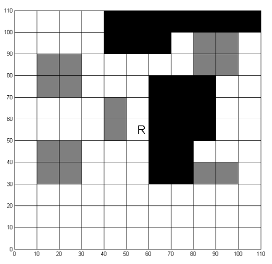
It is evident that 10 pursuers in the grid, very close to the robot and in combination with the obstacles, can lead to surrounding the robot without escape possibilities. Nevertheless, analysing scenarios with different numbers of pursuers, and in possibly the worst case scenarios, allow a better evaluation of the performance of the proposals in complex environments.


(a) 1 pursuer


(b) 2 pursuers


(c) 3 pursuers


(d) 4 pursuers


(e) 5 pursuers

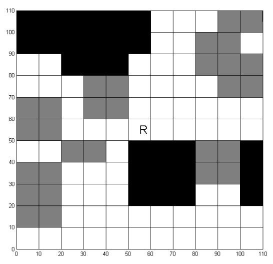

(f) 10 pursuers

Figure 5.2: Examples of approximations for the randomly generated environments of Figure 5.1 into occupancy grids.

Pursuers

For the simulations presented in this chapter, follower pursuers with three different velocities were implemented. A follower pursuer is an entity that tries to get closer to the robot trying to escape, also avoiding collision with obstacles. Thus, the next navigation movement of a pursuer is defined as

$$v_{j+1} = \min_{(a,b)\in\mathcal{A}} d(\text{robot}, p) \tag{5.1}$$

where $\mathcal{A}$ is the set of possible actions for the pursuer, translated as nine possible motion actions to its adjacent cells, (a, b); and $d(\text{robot}, p)$ is the Euclidean distance between the robot and the pursuer (considering their centroids as their current position). The possible motion actions are: right, left, front, back, front right, front left, back right, back left and staying in the same location.

The next sections present the results of all the series of simulations performed to evaluate the proposals described in the previous chapter.

5.2 Guided proteanism, low resolution grid, no updates

The simulations presented in this section have the following objectives:

1. Evaluating the generation of paths that reach the computed local goal (i.e. that would

reach a location away from the pursuers, and possibly a hideaway from some of the pursuers), without falling in local minima of any sort.

2. Analysing which potential function, from all the analysed options (mentioned in Chapter 4), reaches the local goal more successfully depending on the number of pursuers in the environment.
3. Analysing the effect of the subgoals (*a priori* proteanism) in computing different paths to the ones resulting from traditional steepest descent. It is expected that using smaller values of the radius to compute the subgoals, r_{sg}, leads to more difference in the two types of paths.
4. Evaluating the effect (positive or negative) when extending the sensing of obstacles and pursuers by predicting if adjacent cells are part of obstacles or pursuers.

The specific parameters used for the simulations in this section, from Table 5.2, correspond to 1000 random repetitions (with 1000 randomly generated environments), and the rest are as indicated for low resolution grids. The robot starts an experiment in the centre of the grid, and it moves from cell to cell. The velocity of the pursuers is not considered in the simulations of this section.

In order to evaluate the differences of the paths, the mean of the discrete Fréchet distance or DFD (Eiter and Mannila, 1994) and the root-mean-square error or RMSE, along with their standard deviations were computed for all the simulations. Both measures are presented in Appendix A. The success of the paths was considered positive if the local goal was reached by a determined path, or negative if not reached, for both kinds of paths. All the simulations produced one steepest descent path, but not necessarily a protean path, so the total success of the steepest descent paths was computed over the total number of simulations. The total success of the protean paths was accounted only for the number of simulations that produced a protean path.

The flow of the computations for the analysis of the environment and path planning used in the simulations of this section is shown in Figure 5.3, a modified version of Figure 4.16 according to the objectives and experiment settings described.

Subsection 5.2.1 shows some visual examples of computed paths for different kinds of potential functions, highlighting the main findings. In Subsection 5.2.2, the similarities in the computed paths, protean (subgoal based) or not (steepest descent only), are evaluated by means of the DFD and RMSE, the results are shown in the form of tables. Subsection 5.2.3 presents the evaluation of the success reaching the local goal, for protean and steepest descent paths using different potential functions, in the form of comparative graphs of the statistics.

5.2.1 Visual examples of computed paths

The following visual examples show examples of paths computed with and without proteanism (subgoals), for 2 pursuers and a radius of computation for the subgoals of $r_{sg} = 2$

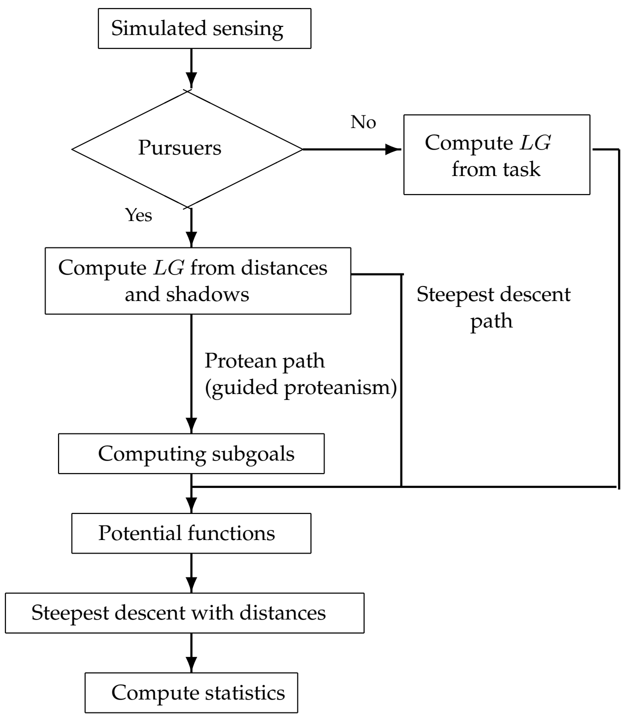
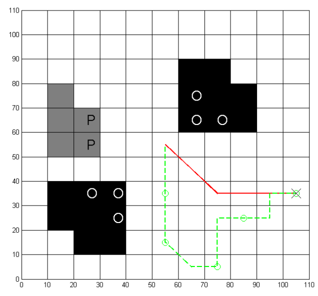
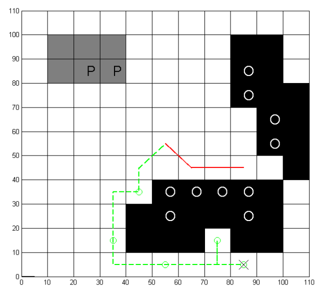


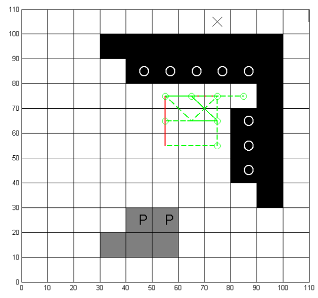
Figure 5.3: Flow diagram of the computations for the simulations to evaluate the success to reach the local goal, for low resolution grids, without updating the sensing from the environment, and calculating the statistics to compare both paths (steepest descent only and subgoal based). By not using the proteanism, a steepest descent only path is computed.



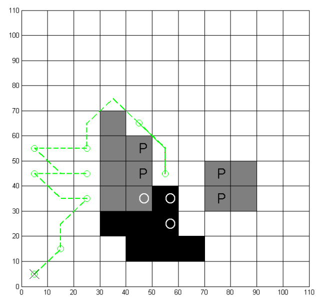
(a) Reaching the local goal



(b) Local minima in steepest descent path



(c) Oscillations due to enclosure



(d) Local minima in steepest descent path

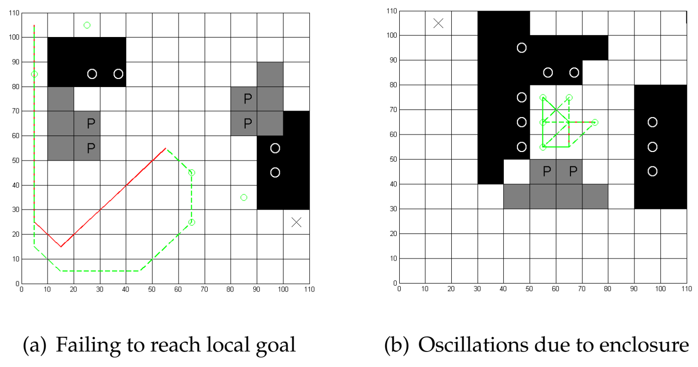
Figure 5.4: Computed paths using steepest descent (solid red), and updated subgoals and steepest descent (dashed green), with subgoals as circles, and discrete potential functions; sensed obstacles in black with an O, sensed pursuers in grey with a P, extended probable obstacles in black, extended probable pursuers in grey, and the LG as X.

navigation steps. Figure 5.4 shows paths computed using modified FIRAS and quadratic goal discrete potential functions. Figures 5.5 and 5.6 show examples using a BVP for the Laplace equation, solved with Jacobi and SOR iterative methods, respectively. Figure 5.7 shows some examples computed with wave-front potential functions. More examples are provided in Appendix B.

The discrete potential functions from modified FIRAS function and a quadratic goal presented local minima problems when used without the subgoals, as expected according to the reports from the literature. The wave-front potential functions also revealed local minima problems due to the geometry of the obstacles when the subgoals are not being used. The use of subgoals evidences an improvement in avoiding local minima, as it causes the change of the potential values of the whole grid, guiding the navigation more smoothly around obstacles.

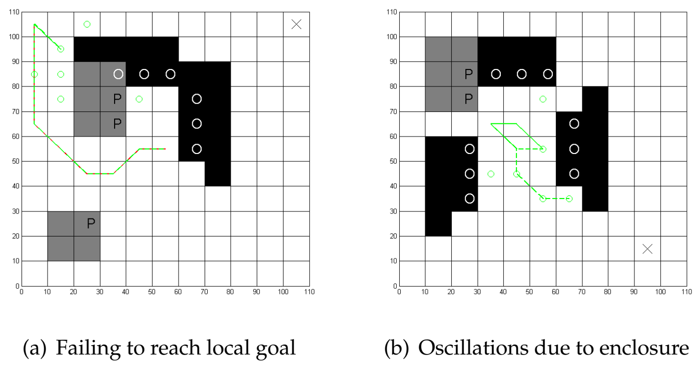
Another evident problem is the oscillation of the paths when enclosed by pursuers and obstacles, as it might not be possible to escape out of the enclosure even with the subgoals. But this problem is not expected from the start in realistic scenarios.

The final problem encountered after a visual inspection is that some paths computed from a BVP for any of both equations, Laplace or perturbation, reach the local goal and then



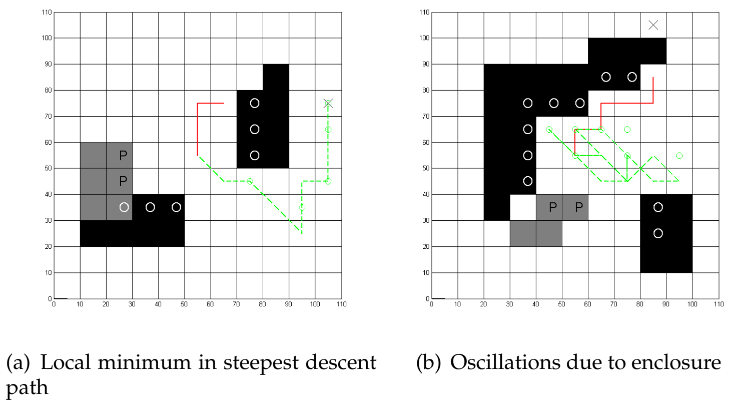
(a) Failing to reach local goal (b) Oscillations due to enclosure

Figure 5.5: Computed paths using steepest descent (solid red) and updated subgoals and steepest descent (dashed green), with subgoals as circles, and a BVP for Laplace's equation with Dirichlet conditions and Jacobi iterative method; sensed obstacles in black with an O, sensed pursuers in grey with a P, extended probable obstacles in black, extended probable pursuers in grey, and the *LG* as X.



(a) Failing to reach local goal (b) Oscillations due to enclosure

Figure 5.6: Computed paths using steepest descent (solid red), and updated subgoals and steepest descent (dashed green), with subgoals as circles, and a BVP for Laplace's equation with Dirichlet conditions and SOR iterative method; sensed obstacles in black with an O, sensed pursuers in grey with a P, extended probable obstacles in black, extended probable pursuers in grey, and the *LG* as X.



(a) Local minimum in steepest descent path (b) Oscillations due to enclosure

Figure 5.7: Computed paths using steepest descent (solid red), and updated subgoals and steepest descent (dashed green), with subgoals as circles, and wave-front potential functions; sensed obstacles in black with an O, sensed pursuers in grey with a P, extended probable obstacles in black, extended probable pursuers in grey, and the *LG* as X.

continue moving away. A criterion that enforces a stop in the computation of the path after reaching the local goal is needed for the use of these potential functions, to save some time in the total deliberation. This problem becomes less important as the path would be recomputed constantly with the updating of the sensing in a real application of the algorithms.

5.2.2 Differences in guided protean paths and steepest descent only paths

In this subsection, the differences in the guided protean path with respect to a path computed towards a local goal were measured statistically from two criteria: the DFD, a measure of the similarity between discrete curves (Eiter and Mannila, 1994), and the RMSE, a measure of the dispersion over a guideline. A maximum of 10 subgoals were allowed in the computation of a protean path, and all the other parameters correspond to the mentioned in Table 5.2 and the beginning of the section, for low grid resolutions. The tables corresponding to the different used potential functions are included in Appendix D.

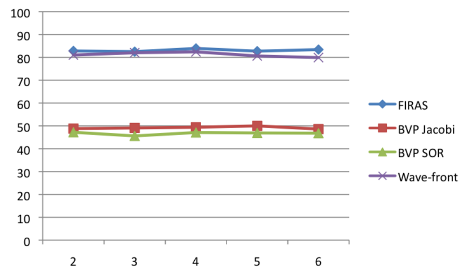
The values of DFD and RMSE in Tables D.1 to D.6 in Appendix D, decrease from left to right according to the radius of computation of the subgoals, r_{sg}, being more evident for environments with less pursuers and using approximations to harmonic equations. This means that the protean paths are significantly different to the ones computed with steepest descent when the subgoals are computed using a smaller radius r_{sg}. Commuting the subgoals closely in terms of navigation steps means increasing the randomness in the path. Longer commutations (larger r_{sg}) make the subgoal based path converge into a steepest descent path. The results are similar with and without extending the sensed obstacles and pursuers to the adjacent cells.

5.2.3 Success computing paths that reached the local goal

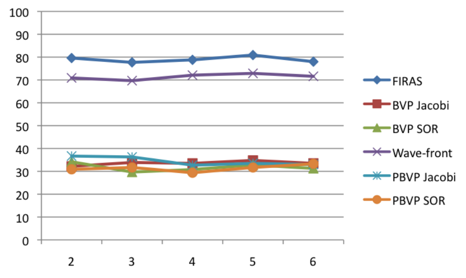
This section presents the results of the relative success of the paths reaching the local goal, computed as mentioned previously, with parameters from Table 5.2 for simulations considering low resolution grids and without updating the sensing of the environment (the speed of the pursuers is not relevant).

Figures 5.8 and 5.9 show the success of steepest descent based and protean or subgoal based paths, correspondingly, when the identified elements in the environment are extended to adjacent cells. Figures 5.10 and 5.11 show the success when the elements in the environment are not extended to adjacent cells, for the same kinds of paths. The legends in the plots correspond to the following potential functions: FIRAS to the FIRAS and quadratic goal potential functions, BVP Jacobi and BVP SOR to the solution to the Laplace equation by a BVP and Jacobi or SOR iterative methods, PBVP Jacobi and PBVP SOR to the solution to the Laplace with perturbation equation by a BVP and Jacobi or SOR iterative methods, and Wave-front to wave-front potential functions.

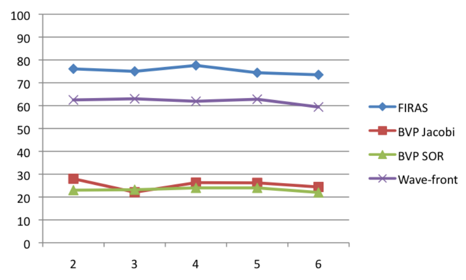
The graphs from Figures 5.8, 5.9, 5.10 and 5.11 indicate that the best performing potential



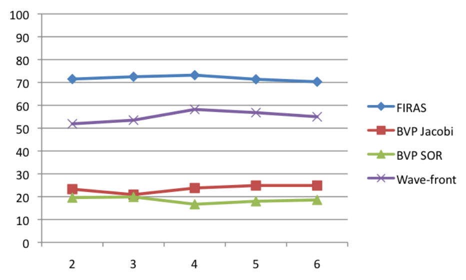
(a) 1 pursuer



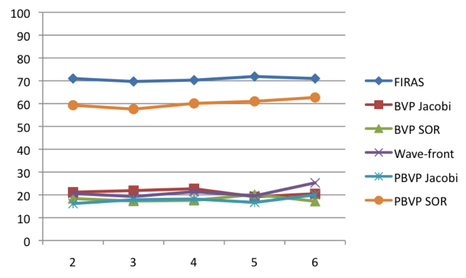
(b) 2 pursuers



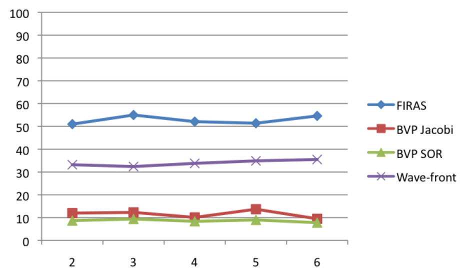
(c) 3 pursuers



(d) 4 pursuers

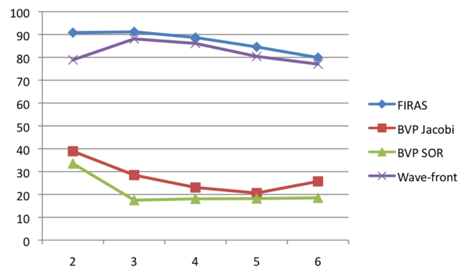


(e) 5 pursuers

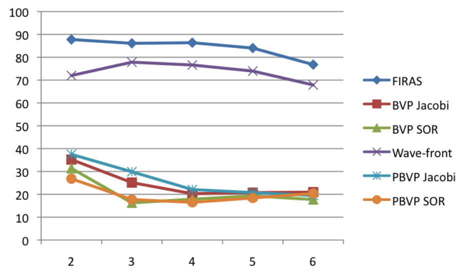


(f) 10 pursuers

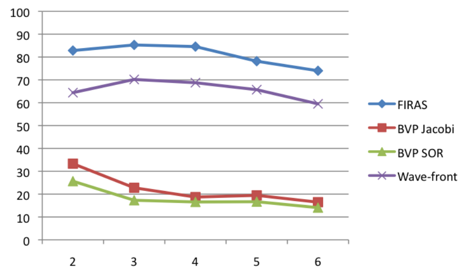
Figure 5.8: Relative success (in percentage) of steepest descent based paths and different potential functions reaching the local goal, when varying the radius of computation of subgoals for protean paths. Obstacles and pursuers are extended to adjacent cells.



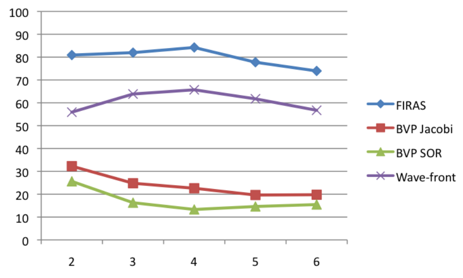
(a) 1 pursuer



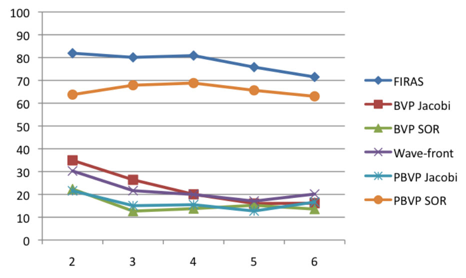
(b) 2 pursuers



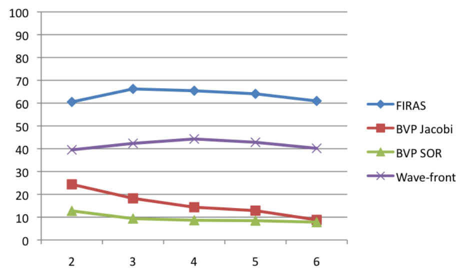
(c) 3 pursuers



(d) 4 pursuers

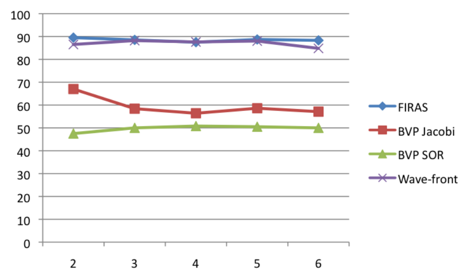


(e) 5 pursuers

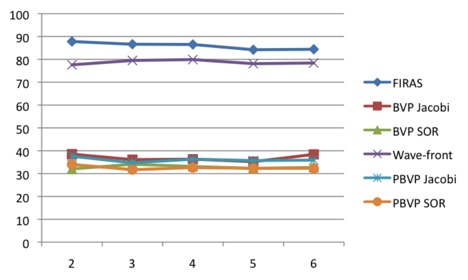


(f) 10 pursuers

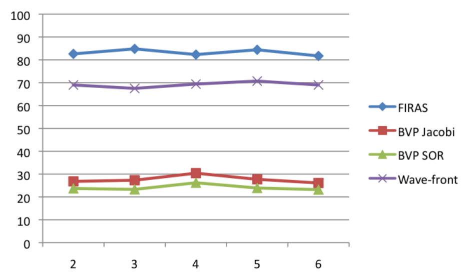
Figure 5.9: Relative success (in percentage) of subgoal based paths (protean) and different potential functions reaching the local goal, when varying the radius of computation of subgoals for protean paths. Obstacles and pursuers are extended to adjacent cells.



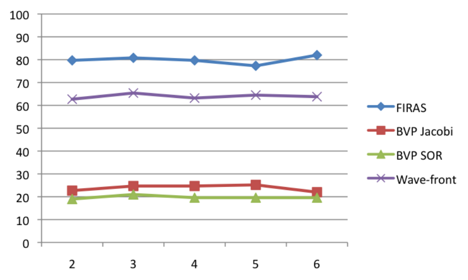
(a) 1 pursuer



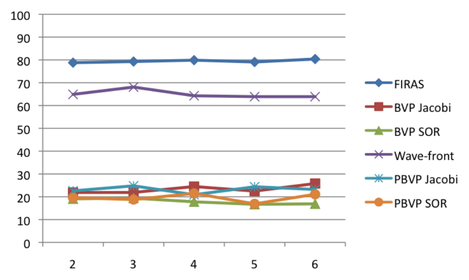
(b) 2 pursuers



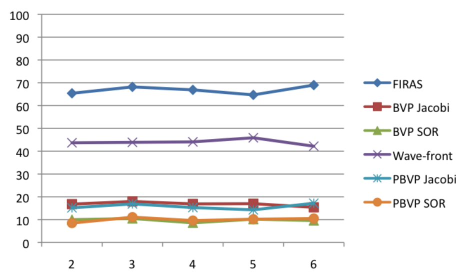
(c) 3 pursuers



(d) 4 pursuers

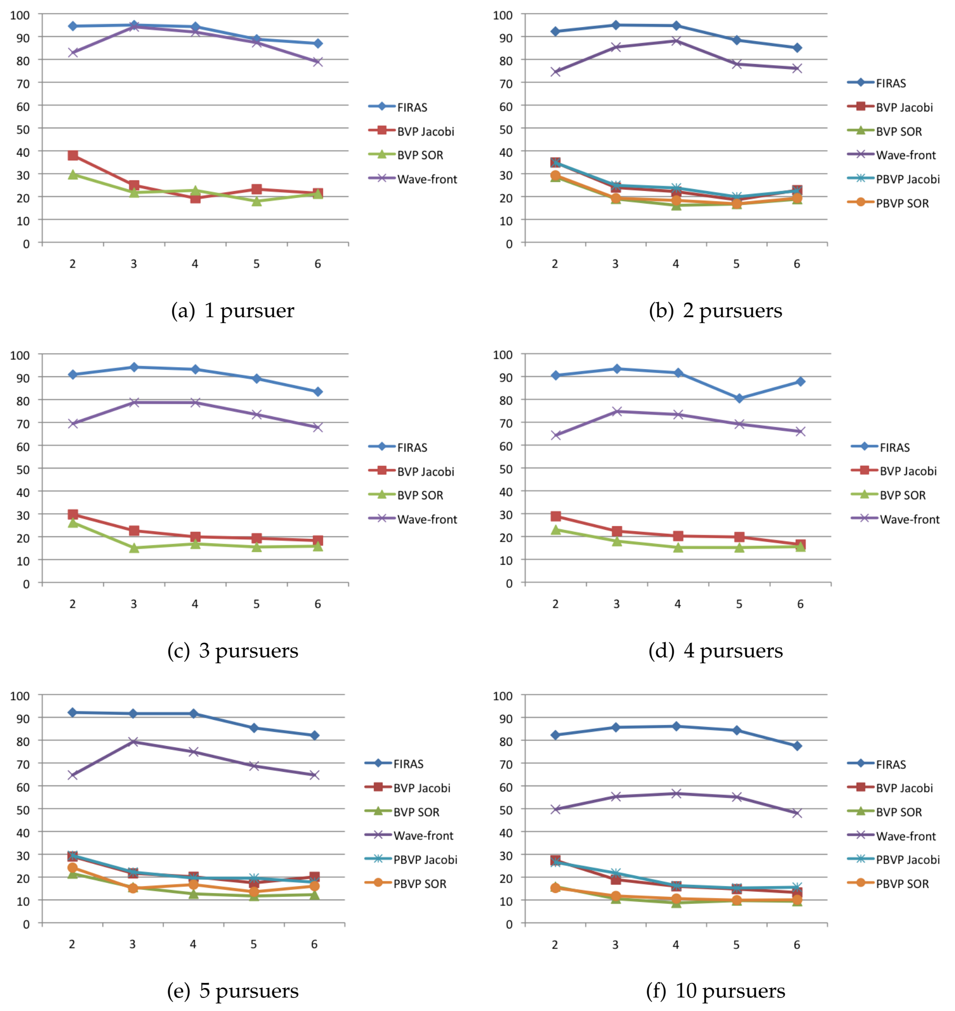


(e) 5 pursuers



(f) 10 pursuers

Figure 5.10: Relative success (in percentage) of steepest descent based paths and different potential functions reaching the local goal, when varying the radius of computation of subgoals for protean paths. Obstacles and pursuers are not extended to adjacent cells.



(a) 1 pursuer

(b) 2 pursuers

(c) 3 pursuers

(d) 4 pursuers

(e) 5 pursuers

(f) 10 pursuers

Figure 5.11: Relative success (in percentage) of subgoal based paths (protean) and different potential functions reaching the local goal, when varying the radius of computation of subgoals for protean paths. Obstacles and pursuers are not extended to adjacent cells.

functions are the modified FIRAS and quadratic goal, followed closely by the wave-front functions. This result is consistent for all the different numbers of pursuers in the simulations. The success of the steepest descent based paths is approximately constant for the different values of r_{sg}, as shown in each one of the individual plots in Figures 5.8 and 5.10.

Figures 5.9 and 5.11 show that the best values for the radius to compute the protean paths are of 2 or 3 cells in the grid, consistent for the different potential functions. It is possible to observe that the subgoal based paths perform better than the steepest descent only paths in reaching the local goal, as the randomness of their computation prevents falling and staying in local minima.

The extension of the sensed pursuers and obstacles in the grid to adjacent cells according to predictions was found to cause more local minima problems, as it can be observed from the difference in the success between Figures 5.8 and 5.10, and Figures 5.9 and 5.10. The success is reduced when extending the environments according to predictions. The extension of elements in the environment reduces the available free space, and thus reducing the possibilities of escaping from enclosures formed by pursuers and obstacles. The problem became more evident with the largest number of pursuers (10), due to the formation of more enclosures in the grid.

5.3 Guided proteanism, low resolution grid, sensing updates

The simulations presented in this section have the following objectives:

1. Evaluating if the proposed approaches (the environment analysis, and the steepest descent only and subgoal based path) prevent the capture of a robot in the window of time of the test (20 navigation steps), for different velocities of pursuers looking to capture greedily.
2. Analysing which one of the potential functions prevents the capture more successfully for the different numbers of pursuers and velocities.
3. Evaluating the effect of the updates of the sensing (measured in navigation steps) in the prevention of capture by pursuers with different velocities.

The success of a path with respect to the other is considered if any of two conditions occur: if the robot is captured some navigation steps later compared to the other path, or if the robot is not captured at all. The average of the capture time (measured in navigation steps) for 100 randomly generated environments is computed as another form to evaluate the performance of the proposals. The simulations have been performed for different numbers of pursuers with velocities of $v_p = v_r, 2v_r, \frac{1}{2}v_r$, different times of updating the environment sensing and a low resolution grid, as specified in Table 5.2.

The flow of the computations for the simulations in this section is shown in Figure 5.12.

Subsection 5.3.1 shows an example of the computation of both paths, steepest descent and protean. Subsection 5.3.2 presents the evaluation of the success avoiding capture, in the

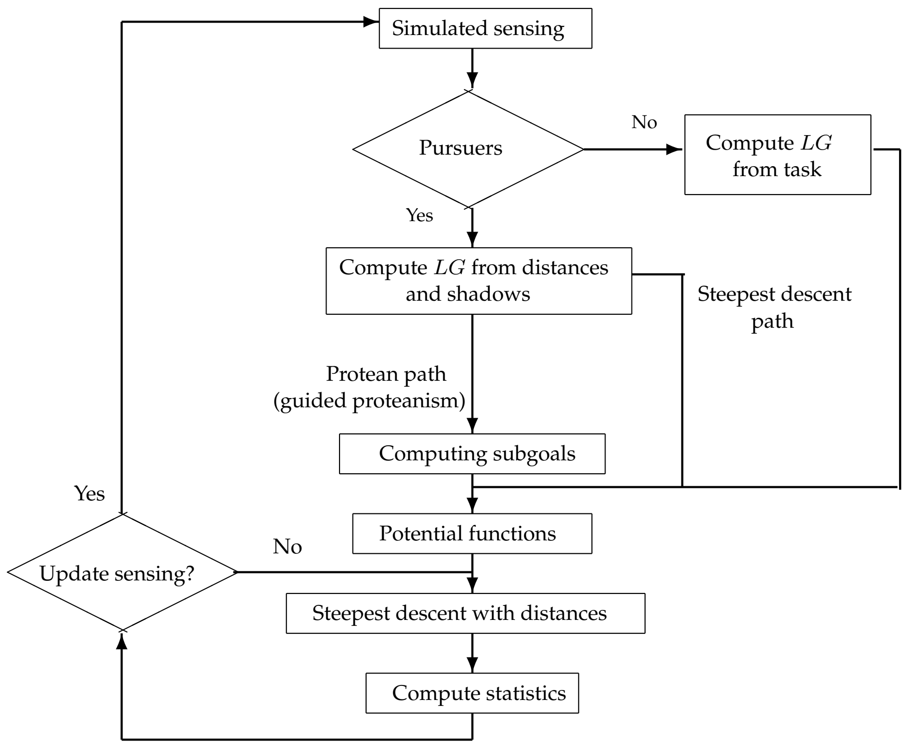


Figure 5.12: Flow diagram of the computations for the simulations to evaluate the success avoiding capture while pursuers move, for low resolution grids, and updating the sensing. Both paths, steepest descent only and subgoal based (protean) are computed independently in the simulations, but following the flow of processing shown in the diagram, as the consequences of the navigation strategies in terms of navigation and motion of the pursuers are different.

form of comparative graphs of the statistics.

5.3.1 Visual examples of computed paths

This subsection presents an example of a steepest descent path and a subgoal based path computed with modified FIRAS and quadratic goal potential functions, when the environment sensing is updated after every navigation step. The pursuers (two) move according to their own strategies and a velocity of $v_p = v_r$. A radius of $r_{sg} = 2$ navigation steps was used to compute the subgoals. The robot starts a path in the centre of the grid, and other parameters were selected considering Table 5.2 for a low resolution grid.

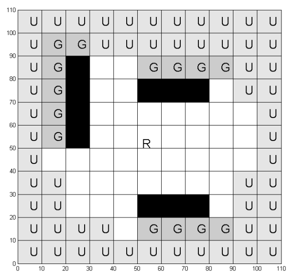
Figure 5.13 presents the first 4 steps of the cycle sensing-path planning, including the sensed environment, and the resulting navigation for the subgoal based path only and the pursuers. The final concentrated paths (including the steepest descent only) are also shown including the navigation of the pursuers. More examples of the concentrated paths, with different velocities for the pursuers, are shown in Appendix B.

5.3.2 Success of computed paths

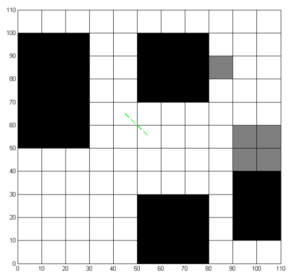
As mentioned before and for both paths, steepest descent only and subgoal based (protean), success occurs when a robot avoids capture in 20 navigation steps, or if the robot is captured some time later than with the other path computed simultaneously. The parameters used in the simulations are as specified in Table 5.2 for low resolution grids and sensing updates at different navigation steps, and pursuers with different speeds.

Significant statistics on the success of the paths are presented in the form of comparative graphs, separated according to different times of updating the sensing (after 1, 3 and 6 navigation steps), the velocity of the pursuers, and then the number of pursuers in the simulations. Figures 5.15 and 5.16 show the results for 1 pursuer, Figures 5.17 and 5.18 for 3 pursuers, and Figures 5.19 and 5.20 for 10 pursuers, respectively, all for updates of the sensing every 1, 3 and 6 navigation steps only. The figures separate the success of the steepest descent based paths from the subgoal based paths accordingly. Other statistics for 2, 4 and 5 pursuers are shown in Appendix E. The legends in the plots correspond to the same potential functions as in previous sections.

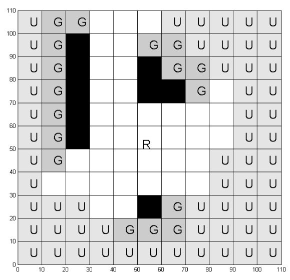
A visual inspection of the graphs of results of this subsection and in Appendix E show that the most successful potential function resulted to be the steepest descent with modified FIRAS and quadratic goal potential functions, followed by steepest descent wave-front. This result holds true for all the numbers and speeds of pursuers and variations in the sensing updates. Nevertheless, the BVP for Laplace equation solved with Jacobi iterative method achieves equivalent success for environments with 4 pursuers. Also, protean paths (subgoal based) become more successful in the interaction with 10 pursuers, compared to steepest descent only paths. A greater variation in the performance of the different



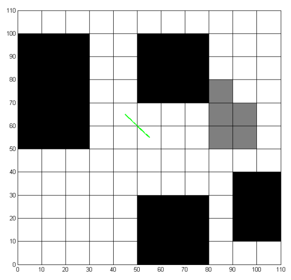
(a) Sensed environment



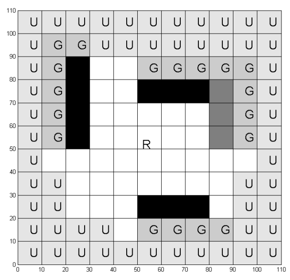
(b) Subgoal based motions



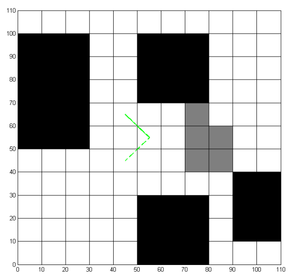
(c) Sensed environment



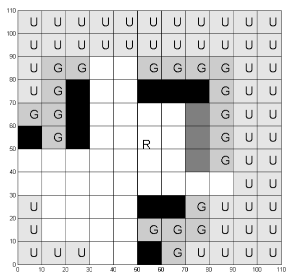
(d) Subgoal based motions



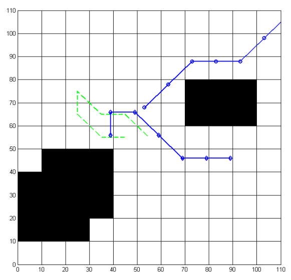
(e) Sensed environment



(f) Subgoal based motions

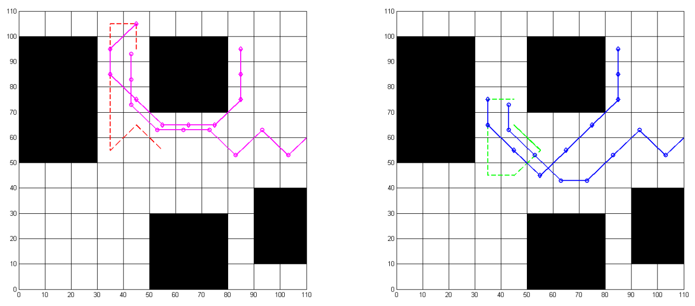


(g) Sensed environment



(h) Subgoal based motions

Figure 5.13: Updates in the sensing and computed respective paths towards the local goal, for the subgoal based path. On the left: sensed obstacles in black, sensed pursuers in dark grey and unknown cells in light grey and a U. On the right: motions in dashed green, sensed obstacles in black, sensed pursuers in grey.



(a) Steepest descent path (b) Subgoal based path

Figure 5.14: Concentrated paths simulating the navigation of the robot and the pursuers with a velocity of $v_p = v_r$, and updating the sensing after every navigation motion. Steepest descent path in red and corresponding pursuers in magenta; subgoal based path in green and corresponding pursuers in blue; obstacles in black.

potential functions occurs when few pursuers are present, and decreases as the number of pursuers grows and when their velocity increases.

The following results arise, considering the results in the previously mentioned graphs (in the section and in the appendices):

- For slower and fewer pursuers (1 or 2), updating the sensing more frequently leads to better results. This can be interpreted as a need to detect the pursuers and to react accordingly, as soon as possible.
- For larger numbers of pursuers (more than 2) and regardless of their speed, updating the sensing frequently does not improve the success. This can be interpreted as the overall reduction of success avoiding capture due to the cluttering of the environment, and consequently a very fast capture is performed by the pursuers regardless of the chosen navigation method of the robot (protean or not).
- For faster and fewer pursuers, the same effect of better results with less frequent updates is observed.

The best radius for computing the subgoal based paths is $r_{sg} = 2$, for all numbers of pursuers, their different speeds, and the potential functions. This is evident from a visual inspection of Figures 5.16, 5.18 and 5.20 showing the subgoal based paths. The proteanism allows reducing unsuccessful paths due to local minima problems from the potential functions, thus avoiding oscillations in the same cell near obstacles and promoting constant motion away from pursuers (through the guidance of the proteanism towards a local goal). A compromise between the radius of the computation of subgoals and the updates of the environment sensing is needed if using protean paths, to achieve the desired effect of apparent randomness in the navigation. For example, updates every 2 navigation steps are not too far in success terms from performing updates every navigation step, and the effect of proteanism in success is greater for a radius of 2 navigation steps. Infrequent updates (e.g., every four navigation steps) could be used when the processing speed is limited, or if the sensed environment has not changed radically and the path is still valid.

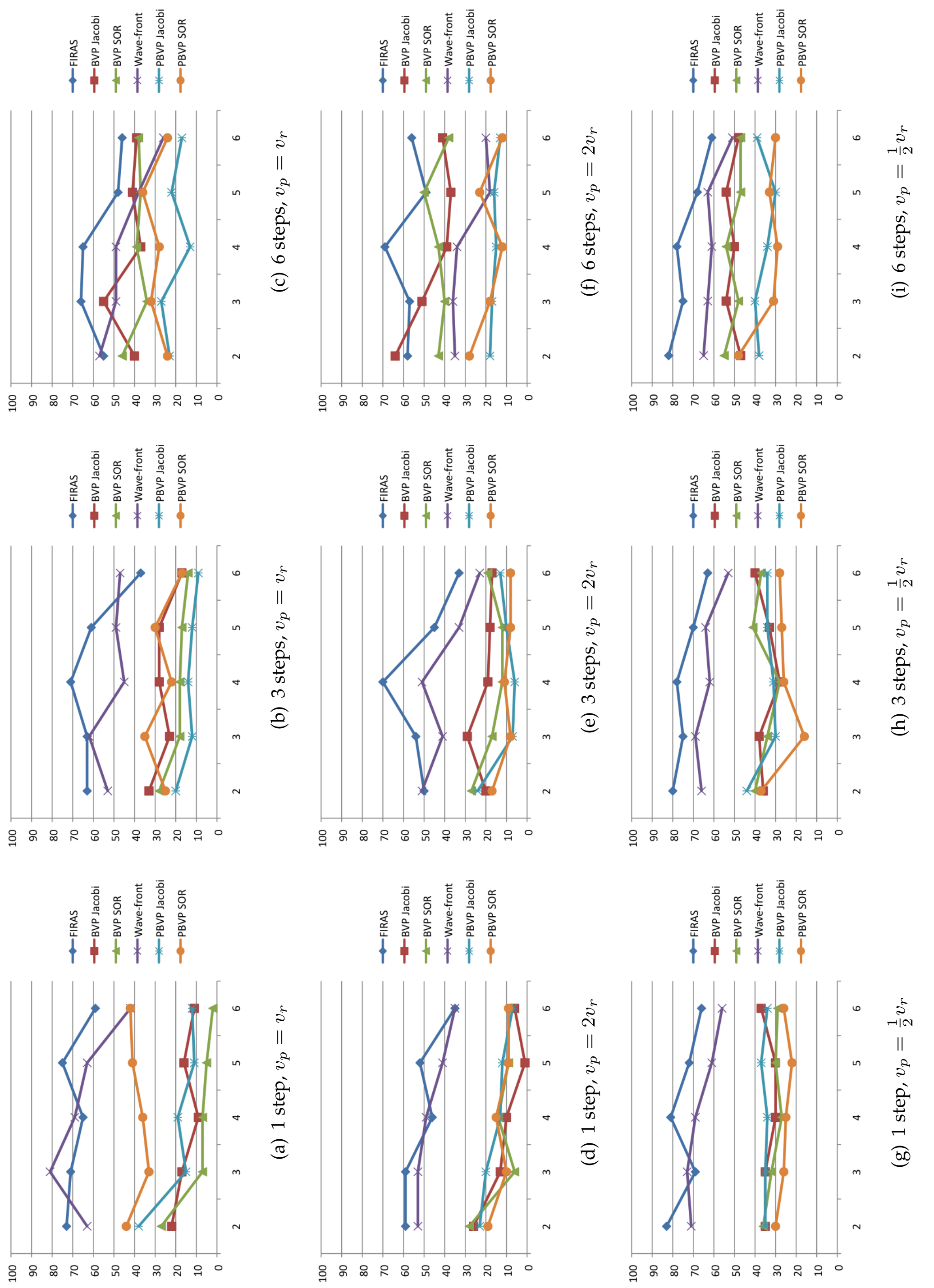
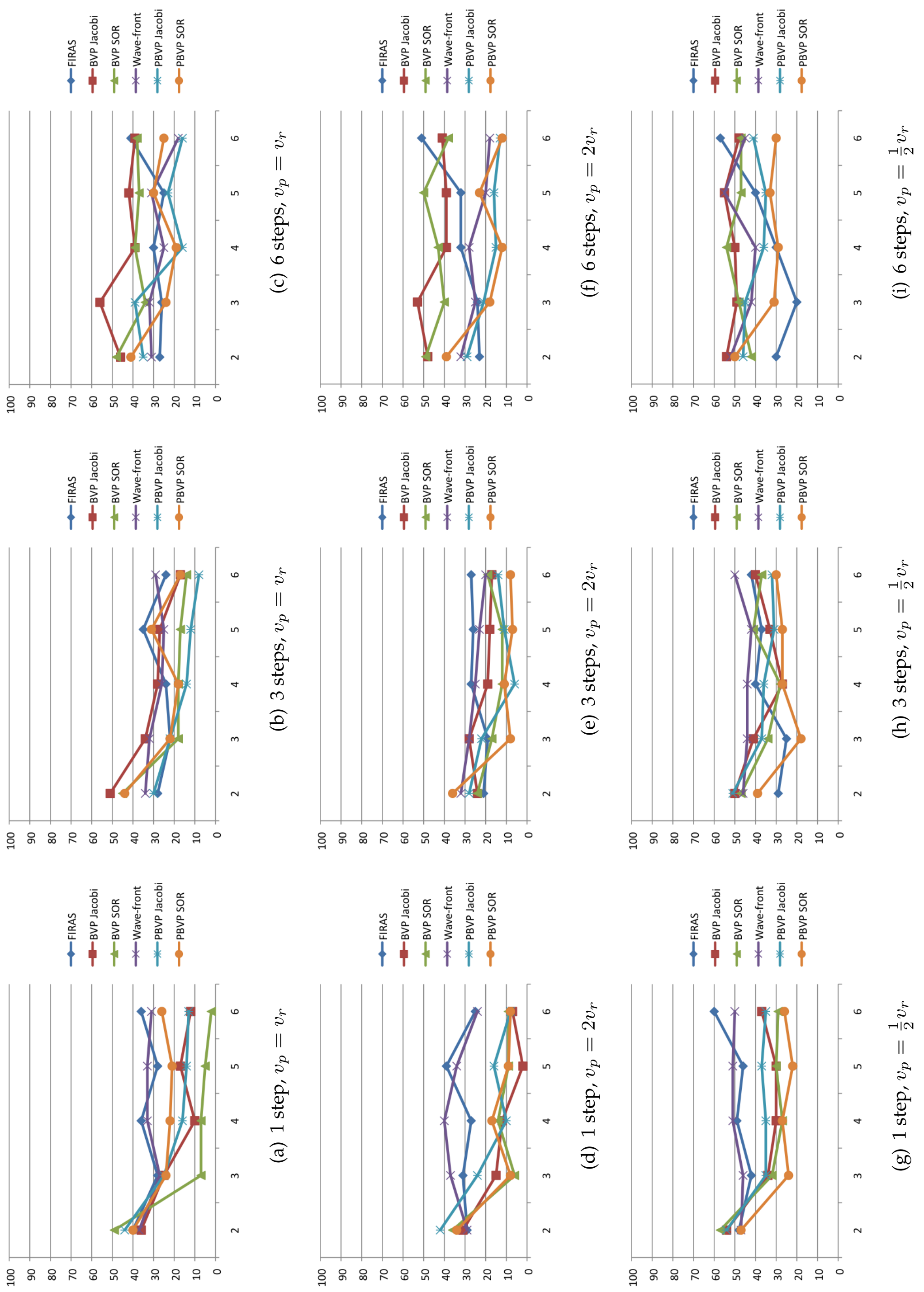


Figure 5.15: Success of steepest descent paths for 1 pursuer, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 1 step, $v_p = v_r$

(b) 3 steps, $v_p = v_r$

(c) 6 steps, $v_p = v_r$

(d) 1 step, $v_p = 2v_r$

(e) 3 steps, $v_p = 2v_r$

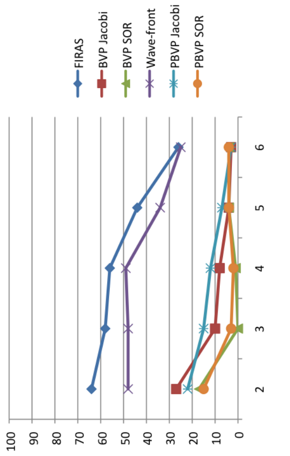
(f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$

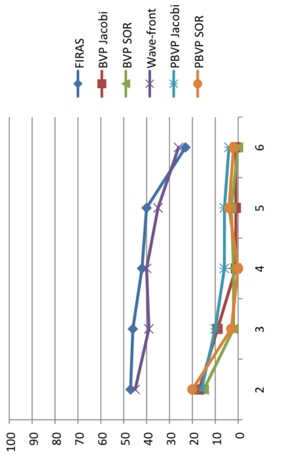
(h) 3 steps, $v_p = \frac{1}{2}v_r$

(i) 6 steps, $v_p = \frac{1}{2}v_r$

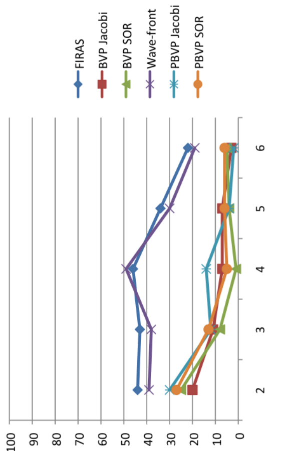
Figure 5.16: Success of subgoal based paths for 1 pursuer, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



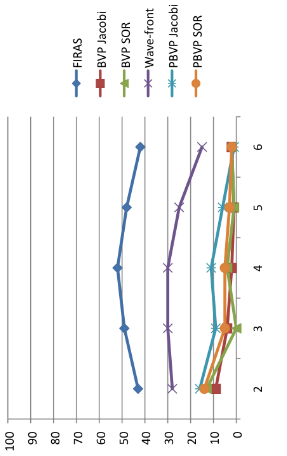
(a) 1 step, $v_p = v_r$



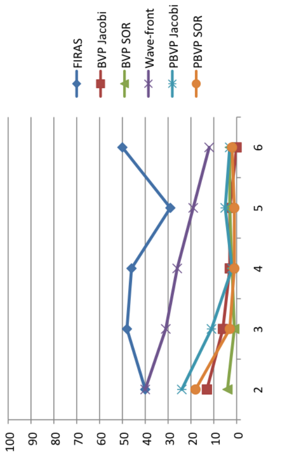
(b) 3 steps, $v_p = v_r$



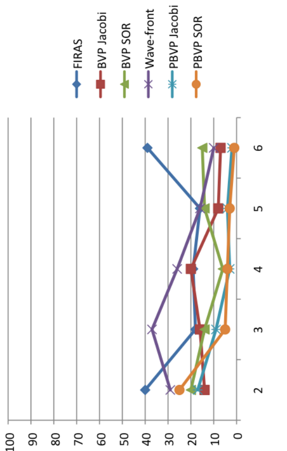
(c) 6 steps, $v_p = v_r$



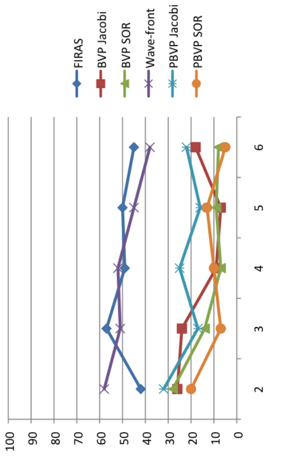
(d) 1 step, $v_p = 2v_r$



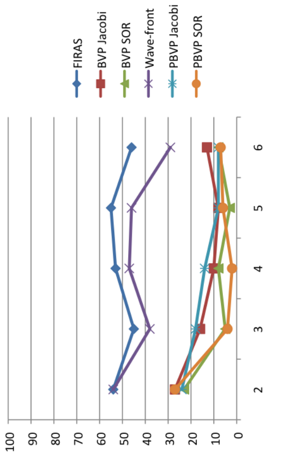
(e) 3 steps, $v_p = 2v_r$



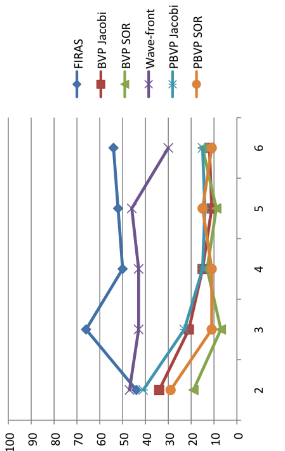
(f) 6 steps, $v_p = 2v_r$



(g) 1 step, $v_p = \frac{1}{2}v_r$

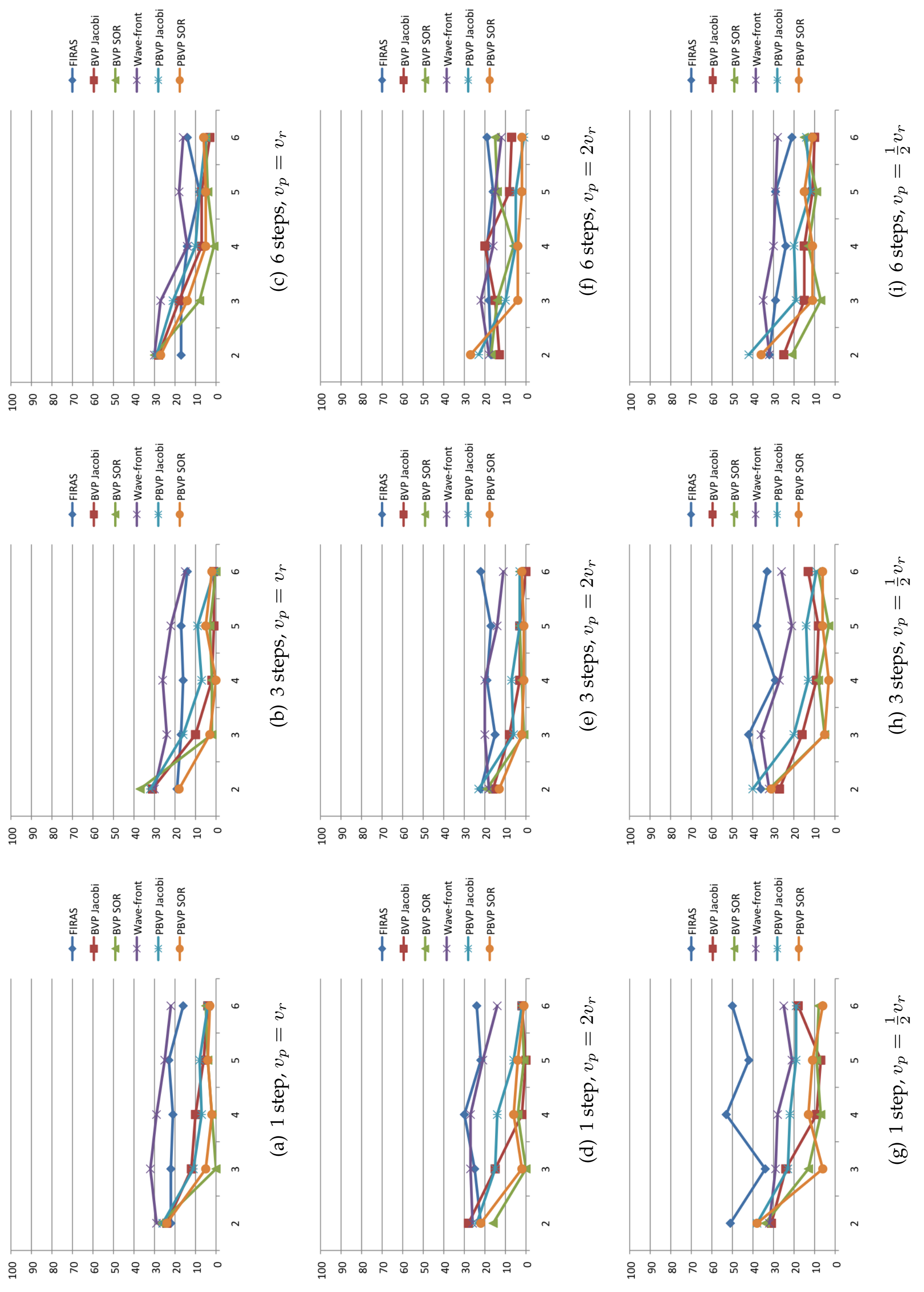


(h) 3 steps, $v_p = \frac{1}{2}v_r$



(i) 6 steps, $v_p = \frac{1}{2}v_r$

Figure 5.17: Success of steepest descent paths for 3 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 1 step, $v_p = v_r$

(b) 3 steps, $v_p = v_r$

(c) 6 steps, $v_p = v_r$

(d) 1 step, $v_p = 2v_r$

(e) 3 steps, $v_p = 2v_r$

(f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$

(h) 3 steps, $v_p = \frac{1}{2}v_r$

(i) 6 steps, $v_p = \frac{1}{2}v_r$

Figure 5.18: Success of subgoal based paths for 3 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.

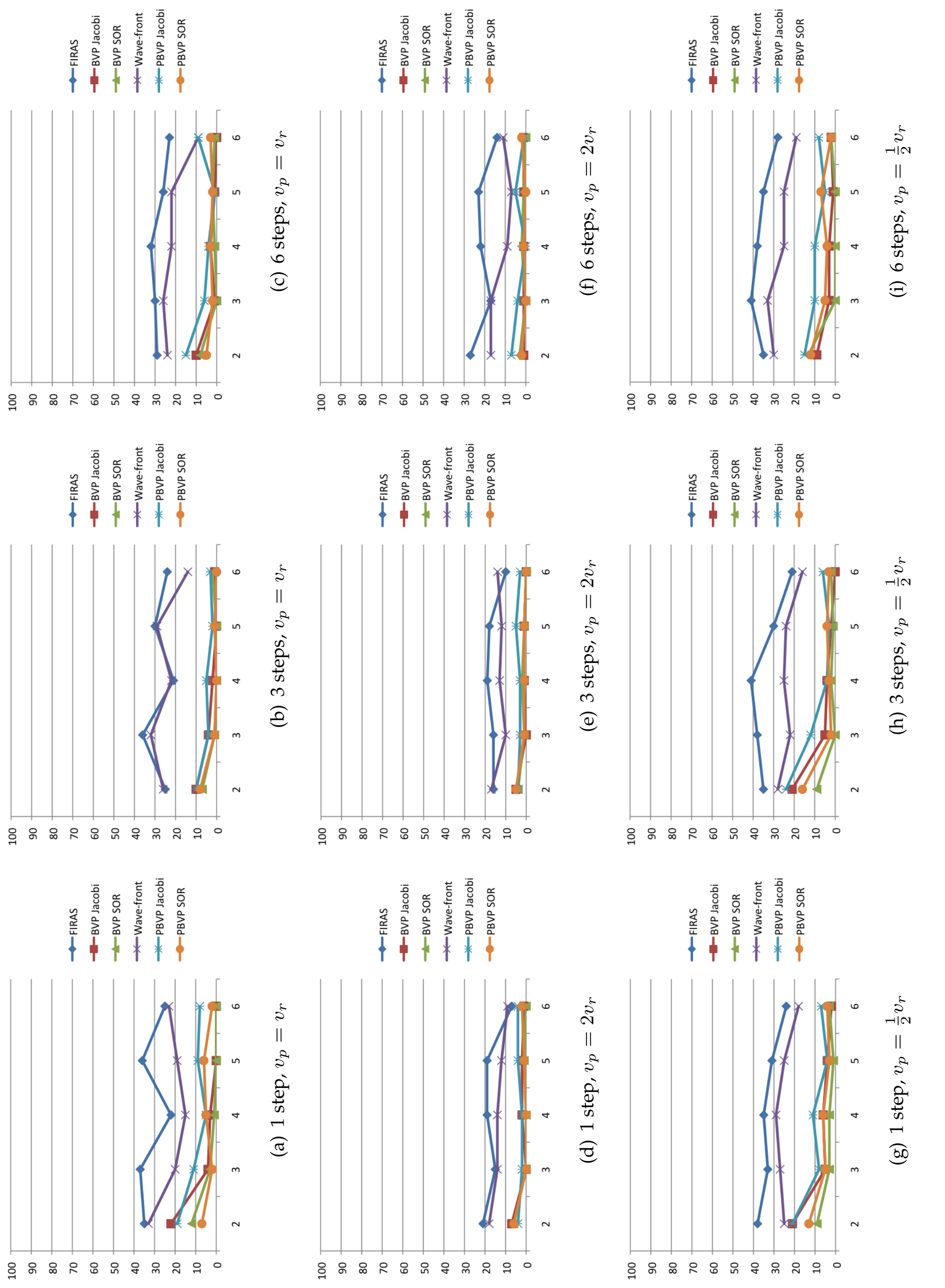


Figure 5.19: Success of steepest descent paths for 10 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.

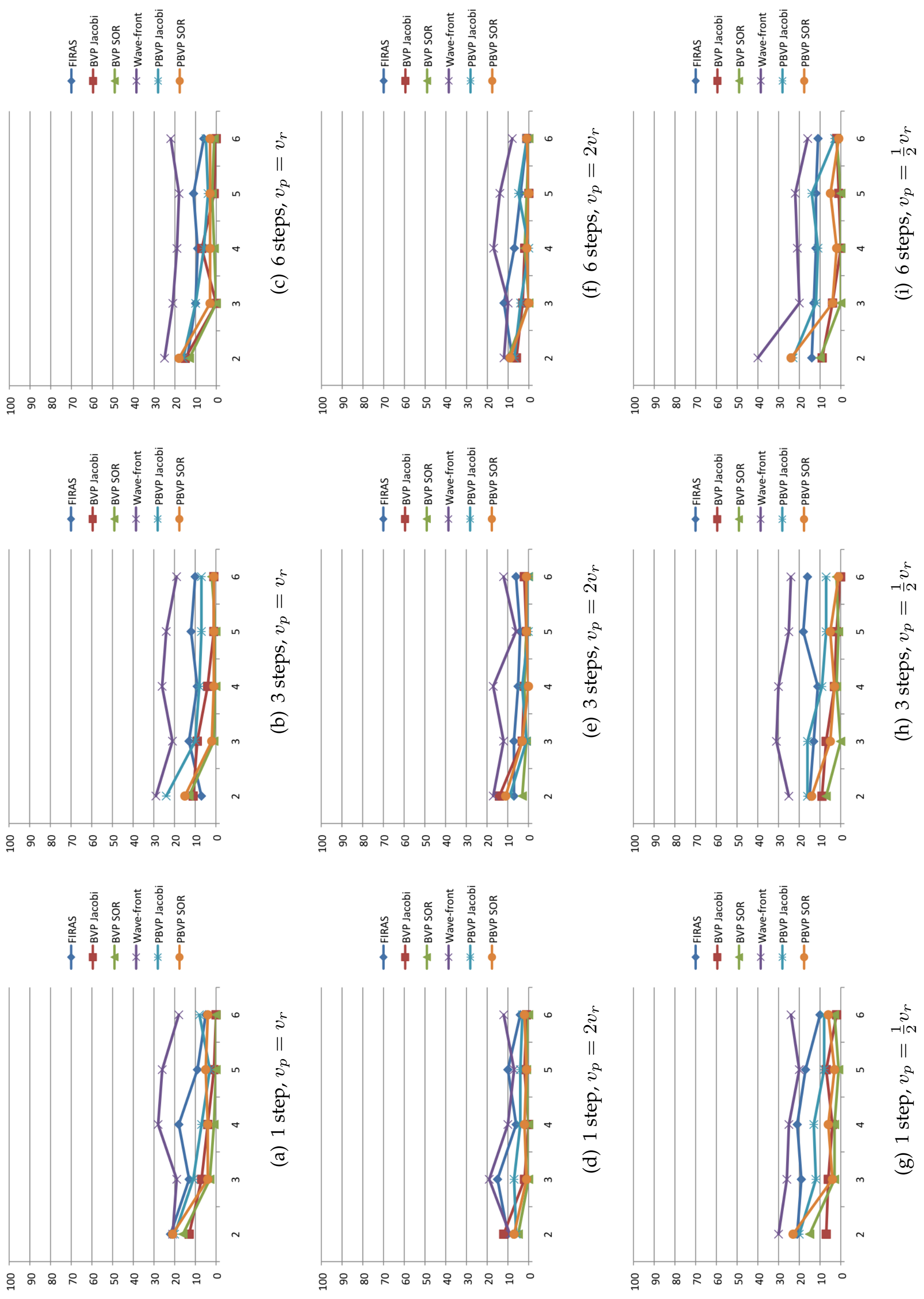


Figure 5.20: Success of subgoal based paths for 10 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.

Statistics of the times of capture (measured as navigation steps for a maximum possible of 20 in an experiment) were computed for the same parameters as in the previous graphs of the subsection. Figures 5.21 to 5.23 present some of the results in the form of box plots that show the distributions of the averages of the time of capture, for the different potential functions. The graphs correspond to the same numbers of pursuers and updates shown previously in Figures 5.15 to 5.19. Subgoal based paths are indicated as SG and steepest descent paths are indicated as SD in the figures, for all the different potential functions. Other box plots are shown in Appendix E.

The results shown in Figures 5.21 to 5.23 confirm the combination of steepest descent with modified FIRAS and quadratic goal potential functions as the best option for the problem of navigation and escape from pursuers, followed by steepest descent and wavefront potential functions. These potential functions allowed the average largest navigation segments before capture, compared to the rest.

The reduction of the chances to escape caused by the increase in the number of pursuers can be observed, as there is less free space to move to in the grid to avoid capture and obstacles simultaneously. When the pursuers are slower, the robot is able to navigate for longer, and the contrary when the pursuers are faster, as proven in visibility based studies of the influence of the velocities in the capture and evasion (Klein and Suni, 2010).

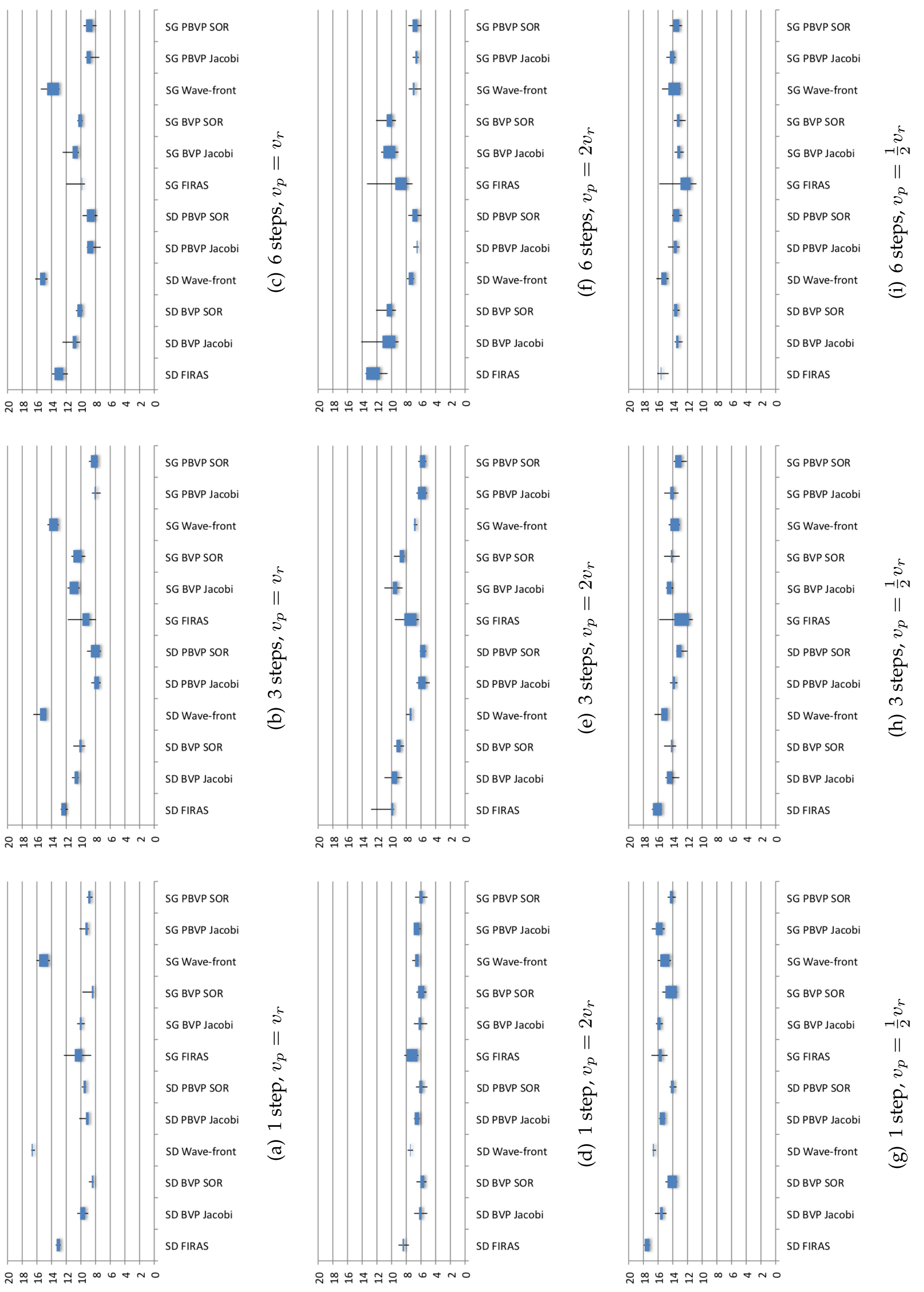
5.4 Guided proteanism, high resolution grid

The main objective of this section is comparing the performance of the proposed environment analysis and path planning computed previously in Section 5.2 when the resolution of the grid is modified (in this case, artificially increased), concerning the reaching of the local goal. The comparisons include the following aspects:

- The best potential function in terms of success reaching the local goal for the different numbers of pursuers.
- If the radius of computation of the subgoals (r_{sg}) has the same effect on the differences of the paths as in the low resolution grids, according to the means of the DFD and RMSE.

The same measures and parameters as in Section 5.2 were used to compute the statistics. Other parameters for the simulations were used as explained in Table 5.2, for high resolution grids and without updating the sensing (the speed of the pursuers is not considered). The robot starts an experiment in the centre of the grid, and it moves from cell to cell. Obstacles and pursuers are not extended to adjacent cells, as results in previous sections indicate that extending the elements does not improve the performance of the paths.

An additional step was incorporated to the processing flow shown in Figure 5.3, where the cells of the sensed elements in the environment are divided and adjusted into a finer grid, as in the dashed parts in diagram of Section 4.6, Figure 4.16.



(a) 1 step, $v_p = v_r$

(b) 3 steps, $v_p = v_r$

(c) 6 steps, $v_p = v_r$

(d) 1 step, $v_p = 2v_r$

(e) 3 steps, $v_p = 2v_r$

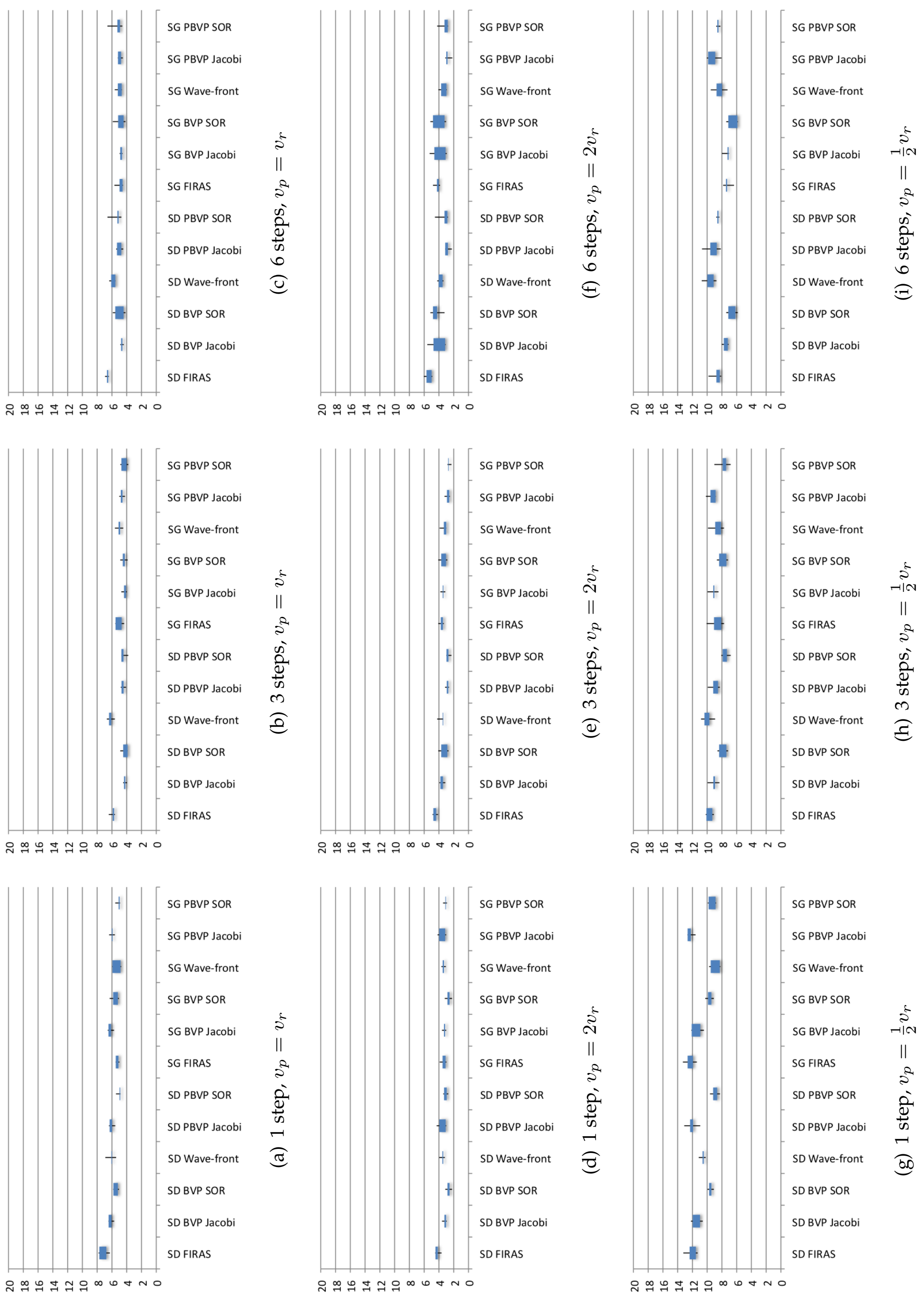
(f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$

(h) 3 steps, $v_p = \frac{1}{2}v_r$

(i) 6 steps, $v_p = \frac{1}{2}v_r$

Figure 5.21: Capture times (in terms of navigation steps) for 1 pursuer, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 100 simulations.



(a) 1 step, $v_p = v_r$

(b) 3 steps, $v_p = v_r$

(c) 6 steps, $v_p = v_r$

(d) 1 step, $v_p = 2v_r$

(e) 3 steps, $v_p = 2v_r$

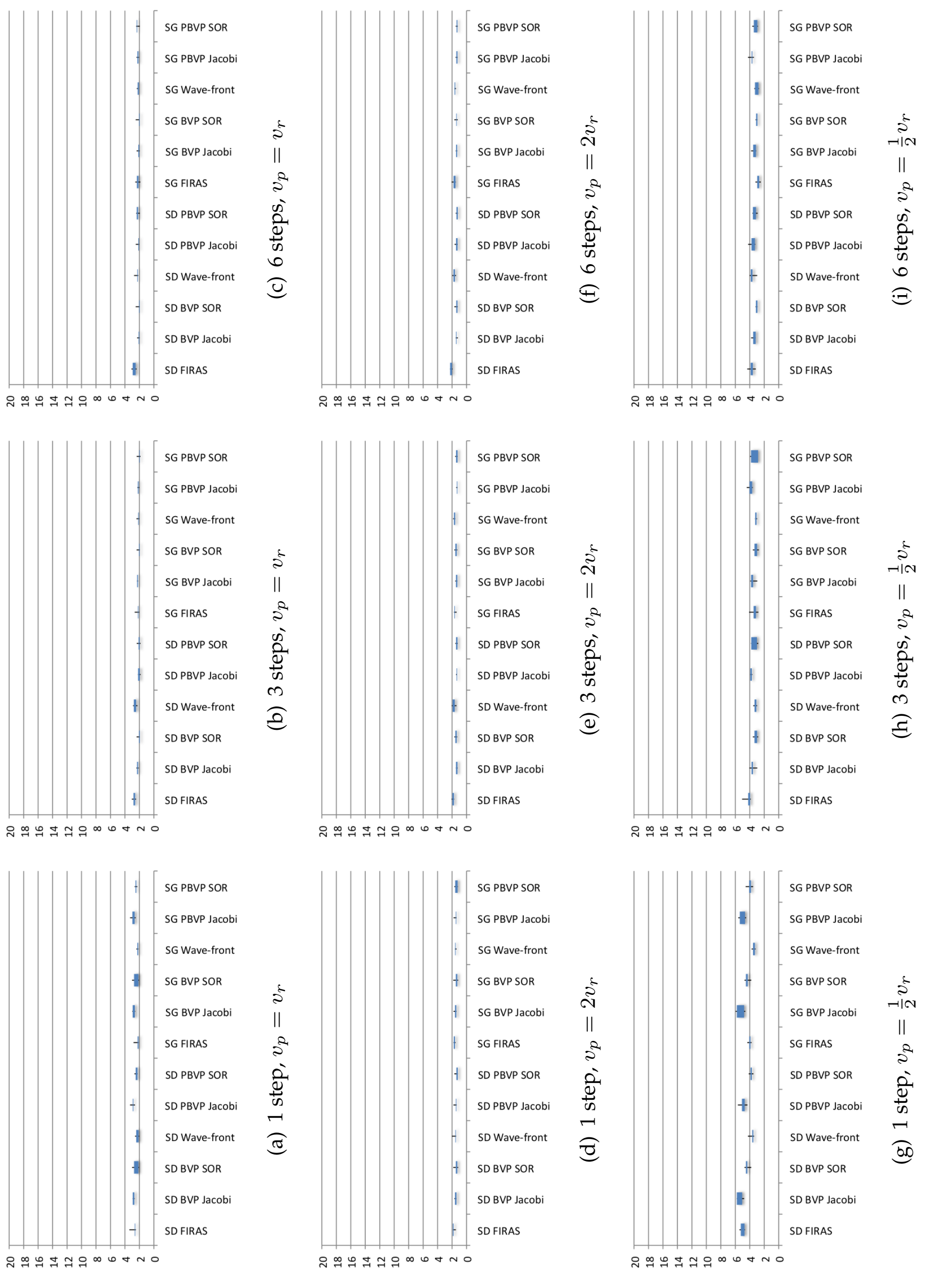
(f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$

(h) 3 steps, $v_p = \frac{1}{2}v_r$

(i) 6 steps, $v_p = \frac{1}{2}v_r$

Figure 5.22: Capture times (in terms of navigation steps) for 3 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 100 simulations.



(a) 1 step, $v_p = v_r$

(b) 3 steps, $v_p = v_r$

(c) 6 steps, $v_p = v_r$

(d) 1 step, $v_p = 2v_r$

(e) 3 steps, $v_p = 2v_r$

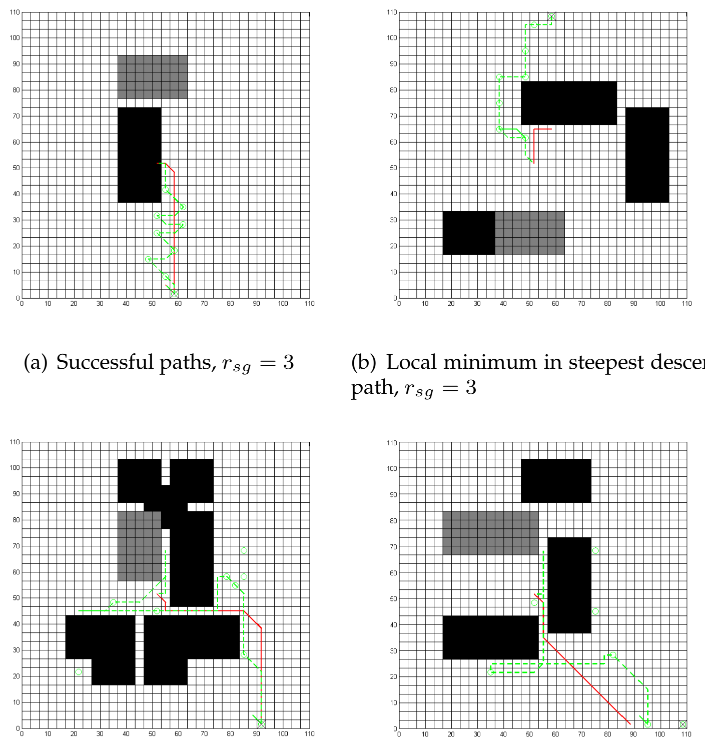
(f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$

(h) 3 steps, $v_p = \frac{1}{2}v_r$

(i) 6 steps, $v_p = \frac{1}{2}v_r$

Figure 5.23: Capture times (in terms of navigation steps) for 10 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 100 simulations.



(a) Successful paths, $r_{sg} = 3$

(b) Local minimum in steepest descent path, $r_{sg} = 3$

(c) Successful paths, $r_{sg} = 9$

(d) Local minima in both paths, $r_{sg} = 9$

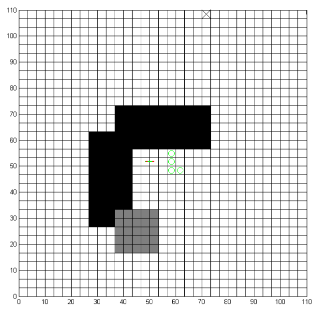
Figure 5.24: Computed paths using steepest descent (solid red), and updated subgoals and steepest descent (dashed green), with subgoals as circles, and discrete potential functions for a high resolution grid; sensed obstacles in black, sensed pursuers in grey, and the LG as X.

Subsection 5.4.1 shows some visual examples of computed paths for different kinds of potential functions, highlighting the main findings. Subsection 5.4.2 presents the statistics of the differences between the steepest descent and the subgoal based paths (measured again through the DFD and RMSE), in the form of tables. Subsection 5.4.3 presents the evaluation of the success of the paths reaching the local goal, in the form of comparative graphs of the statistics.

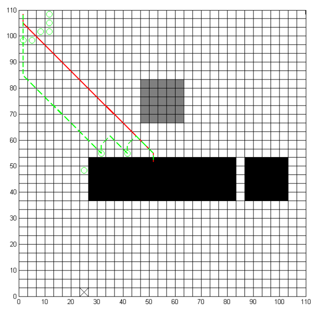
5.4.1 Visual examples of computed paths

This subsection presents visual examples of computed paths towards a local goal, with and without proteanism (subgoals), and for different potential functions and environments. The examples use two pursuers and a radius for the computation of subgoals of r_{sg} = 3, 6 and 9 navigation steps. Other parameters have been specified in Table **??**, for high resolution grids.

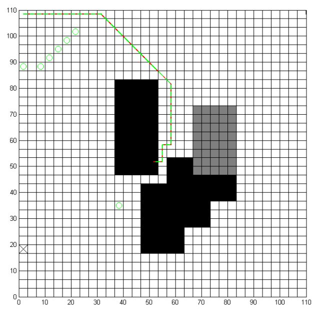
Figure 5.24 shows the resulting paths using modified FIRAS and quadratic goal discrete potential functions, and Figure 5.25 using a BVP for the Laplace equation solved with the Jacobi iterative method. More examples can be found in Appendix C.



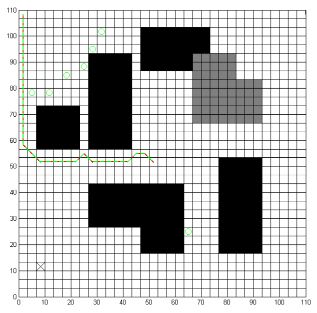
(a) Local minima in both paths, $r_{sg} = 3$



(b) Local minima in both paths, $r_{sg} = 3$



(c) Local minima in both paths, $r_{sg} = 6$



(d) Local minima in both paths, $r_{sg} = 9$

Figure 5.25: Computed paths using steepest descent (solid red), and updated subgoals and steepest descent (dashed green), with subgoals as circles, and a BVP for Laplace equation with Dirichlet conditions and Jacobi iterative method; sensed obstacles in black, sensed pursuers in grey, and the LG as X.

The results presented in the previous figures demonstrate the same problems noted for the low resolution simulations in Section 5.2: the presence of local minima in the steepest descent paths using discrete potential functions (modified FIRAS and quadratic goal) and wave-front expansions. It is also seen that the resulting paths using subgoals have more deviations and navigation loops than the steepest descent only paths.

On the other hand, the paths computed using solutions to the Laplace and Laplace with perturbation equations by a BVP solved with Jacobi iterative methods show problems of equal potential values in the grid for the free space cells (as mathematically there is only a global minimum in the location of the goal). These problematic potential functions cause many of the paths to lead towards a location near an edge of the grid, but not necessarily towards the local goal as expected. It is the same case for both paths, steepest descent and subgoal based.

5.4.2 Differences in guided protean paths and steepest descent only paths

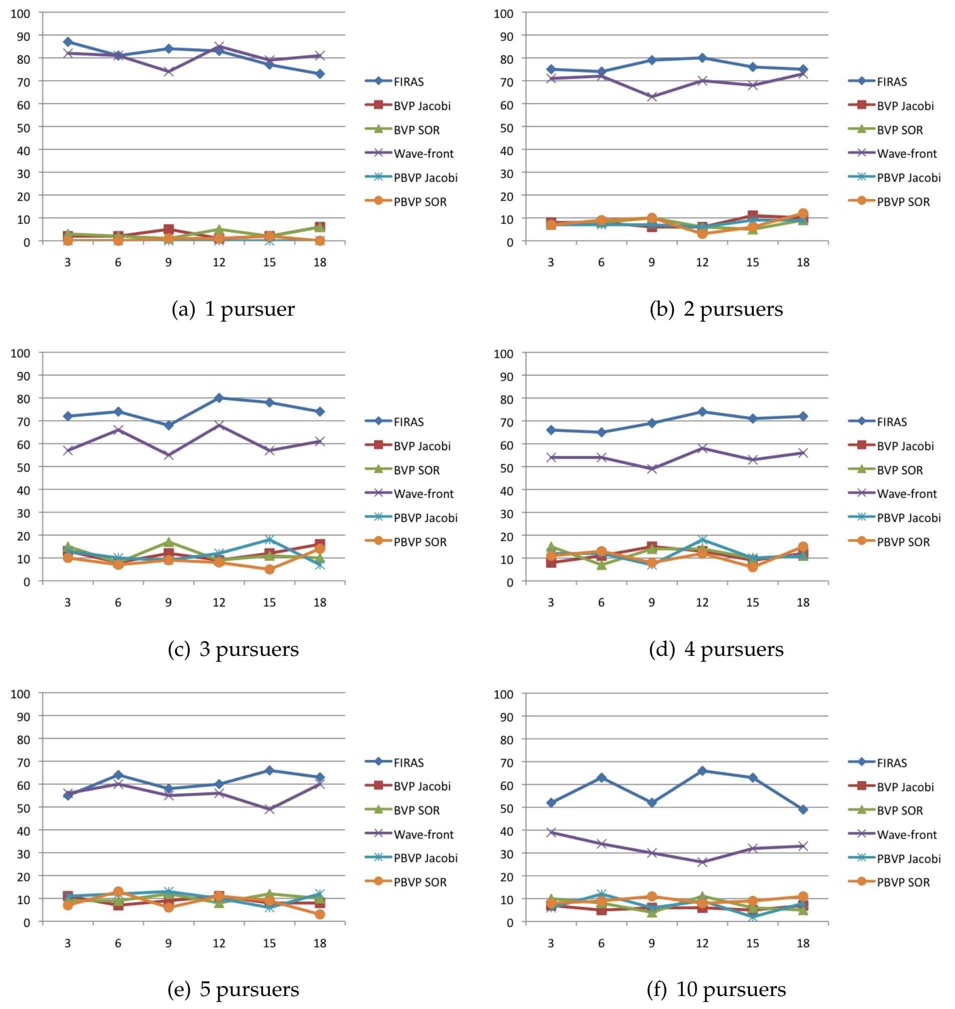
Differences in the guided protean path with respect to a path computed towards a local goal were measured statistically from the same two criteria used for the simulations in Subsection 5.2.2: DFD and RMSE, and a maximum of 30 subgoals for the protean path. The corresponding tables of results have been added in Appendix D. The parameters used to compute the statistics have been presented in Table 5.2, for high resolution grids.

The statistics in the higher resolution grid reinforce the results in Subsection 5.2.3 regarding the effect of the radius of computation of subgoals, as the protean paths are significantly different to the ones computed with steepest descent when the subgoals are computed for a smaller radius. The results for solutions to the Laplace and Laplace with perturbation equations by a BVP and Jacobi or SOR iterative methods indicate the production of almost identical paths for all the values of r_{sg}, as the DFD and RMSE values are comparatively smaller. The tables also confirm that the similarity of the paths increases with the value of r_{sg}, as the subgoal based paths have less randomness in their constitution.

5.4.3 Success of computed paths

This subsection presents the statistical results of the success of the paths trying to reach the local goal, for both protean and steepest descent only paths, and different conditions in the environment. The parameters for the simulations are the same as for the previous subsection, from Table 5.2 for high resolution grids and no updates in the sensing, and the velocity of the pursuers is not relevant in this subsection.

The graphs in Figures 5.26 and 5.27 present the results for the steepest descent paths and the subgoal based paths, correspondingly, using the different potential functions. The legends in the plots correspond to the following potential functions: FIRAS to the FIRAS and quadratic goal potential functions, BVP Jacobi and BVP SOR to the solution to the Laplace equation by a BVP and Jacobi or SOR iterative methods, PBVP Jacobi and PBVP



(a) 1 pursuer

(b) 2 pursuers

(c) 3 pursuers

(d) 4 pursuers

(e) 5 pursuers

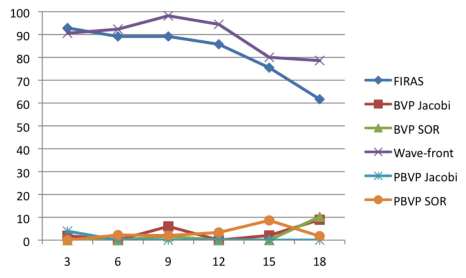
(f) 10 pursuers

Figure 5.26: Plots of the success of reaching the local goal for steepest descent paths using the different kinds of potential functions, in percentage over the total of randomly generated environments (100). The horizontal axis corresponds to the values or r_{sg} and the vertical axis is the percentage of success.

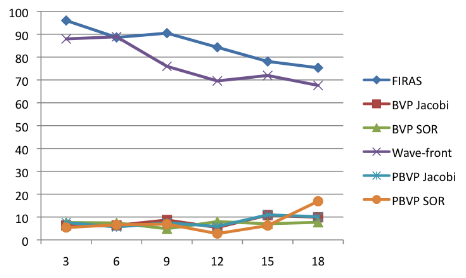
SOR to the solution to the Laplace with perturbation equation by a BVP and Jacobi or SOR iterative methods, and Wave-front to wave-front potential functions.

The same potential functions were found to perform better for all numbers of pursuers as in Subsection 5.2.3: the modified FIRAS and quadratic goal were best, followed closely by the wave-front potential functions. The success of those potential functions is better (according to the percentages) for the higher resolution grid (artificially computed from the original sensing), particularly for the larger numbers or pursuers. The simulations with a higher resolution grid highlighted the differences between the best performing potential functions and the rest, evident in the visible gaps in the plots (Figures 5.26 and 5.27).

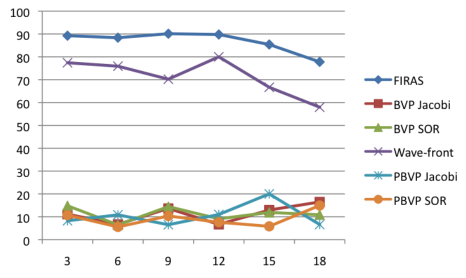
The best radii for the computation of subgoals in combination with the modified FIRAS and quadratic goal potential functions, are 6 and 9 units (measured in the size of a cell in the 33×33 grid), and equivalent to the 2 and 3 found in Subsection 5.2.3 for the 11×11



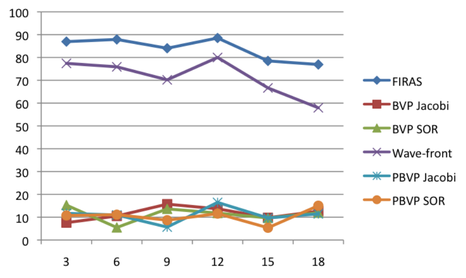
(a) 1 pursuer



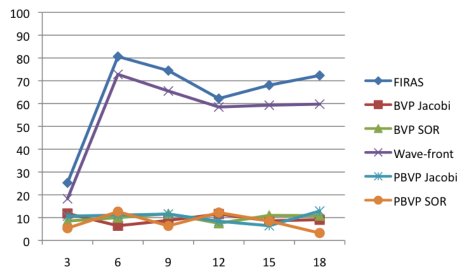
(b) 2 pursuers



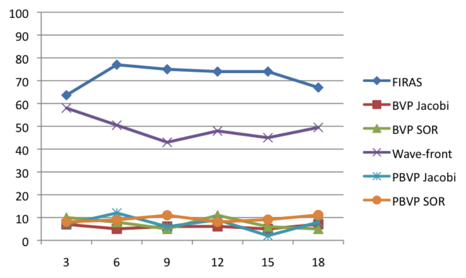
(c) 3 pursuers



(d) 4 pursuers



(e) 5 pursuers



(f) 10 pursuers

Figure 5.27: Plots of the success of reaching the local goal for subgoal based paths using the different kinds of potential functions, in percentage over the total of randomly generated environments (100). The horizontal axis corresponds to the values or r_{sg} and the vertical axis is the percentage of success.

grid. Additionally, the subgoal based paths perform better than the steepest descent only paths as expected, as the randomness allows avoiding local minima problems.

5.5 Guided proteanism, high resolution, sensing updates

The main objective of the simulations in this section is comparing the previous results of success avoiding capture (in Section 5.3), when the resolution of the grid is modified. As before, the number of pursuers and their velocity was varied in the simulations. The environment sensing is updated after an interval of time, and the path towards a local goal is also updated accordingly. The compared aspects include:

- The performance of the potential functions for the different combinations of pursuers.
- The effect of the update timing in the success to avoid capture.

The success of a path with respect to the other is measured as in the simulations in Section 5.3, considering the cases where the robot is not captured in 30 navigation steps, or if a path leads to a latter capture respect to the other simultaneously computed. The parameters for the simulations in this section have been mentioned in Table 5.2, for high resolution grids, sensing updates, and pursuers with velocities of $v_p = v_r, 2v_r, \frac{1}{2}v_r$.

Subsection 5.5.1 shows an example of the computation of both paths, steepest descent and protean. Subsection 5.5.2 presents the evaluation of the success in avoiding capture, in the form of comparative graphs of the statistics.

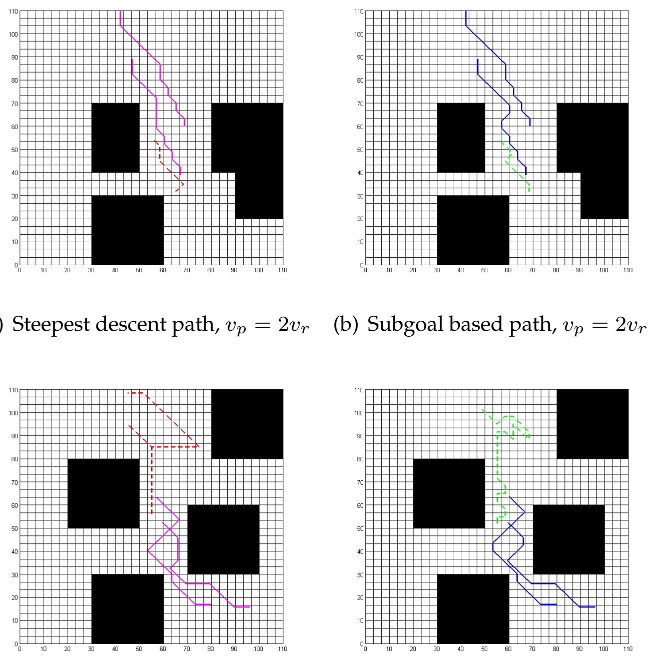
5.5.1 Visual examples of computed paths

This subsection presents some resulting paths computed when the environment sensing is updated, for two pursuers with speeds of $v_p = 2v_r, \frac{1}{2}v_r$, sensing updates every 3 navigation steps, $r_{sg} = 6$ and modified FIRAS and quadratic goal potential functions. Figure 5.28 shows the computed paths. More examples are shown in Appendix C.

5.5.2 Success of computed paths

The success of computed paths avoiding the capture of the robot is presented in this subsection, computed in the same manner as Subsection 5.3.2, but for a maximum of 30 navigation steps. The parameters used in the simulations have been included in Table 5.2, for high resolution grids and sensing updates. The experiments use modified FIRAS and quadratic goal, BVP for the Laplace and Laplace with perturbation equations solved by Jacobi iterative method, and wave-front potential functions.

Significant statistics are presented in the form of comparative graphs, according to different sensing updates (3, 9 and 18 navigation steps), numbers of pursuers, and their velocities. Figures 5.29 and 5.30 show the results for 1 pursuer, Figures 5.31 and 5.31 for 3 pursuers, and Figures 5.33 and 5.34 for 10 pursuers, respectively. The plots have been separated



(a) Steepest descent path, $v_p = 2v_r$ (b) Subgoal based path, $v_p = 2v_r$

(c) Steepest descent path, $v_p = \frac{1}{2}v_r$ (d) Subgoal based path, $v_p = \frac{1}{2}v_r$

Figure 5.28: Paths simulating the navigation of the robot and the pursuers, and updating the sensing after 3 navigation steps, for a higher resolution grid. Steepest descent paths in red and corresponding pursuers in magenta; subgoal based paths in green and corresponding pursuers in blue; obstacles in black.

according to the kind of path computed, steepest descent or subgoal based. Other graphs for 2, 4 and 5 pursuers are shown in Appendix E. The legends in the plots correspond to the same potential functions as in previous sections.

As in the results presented in Section 5.3, the most successful potential function, in general for all examined numbers of pursuers, is the modified FIRAS and quadratic goal, followed by the wave-front one, in combination with the steepest descent computation of the path. In higher resolution grids, the computed success for the different combinations of potential functions and steepest descent or subgoals to compute the paths is quite diverse, compared to the results for low resolution grids. Subgoal based paths, particularly with modified FIRAS and quadratic goal, and wave-front potential functions, improve their overall success compared to low resolution grids.

In the higher resolution grids, the percentages of success are higher than in low resolution grids. This has been interpreted as the availability of more possible motion configurations, and thus the possibility to avoid capture for longer in terms of navigation steps.

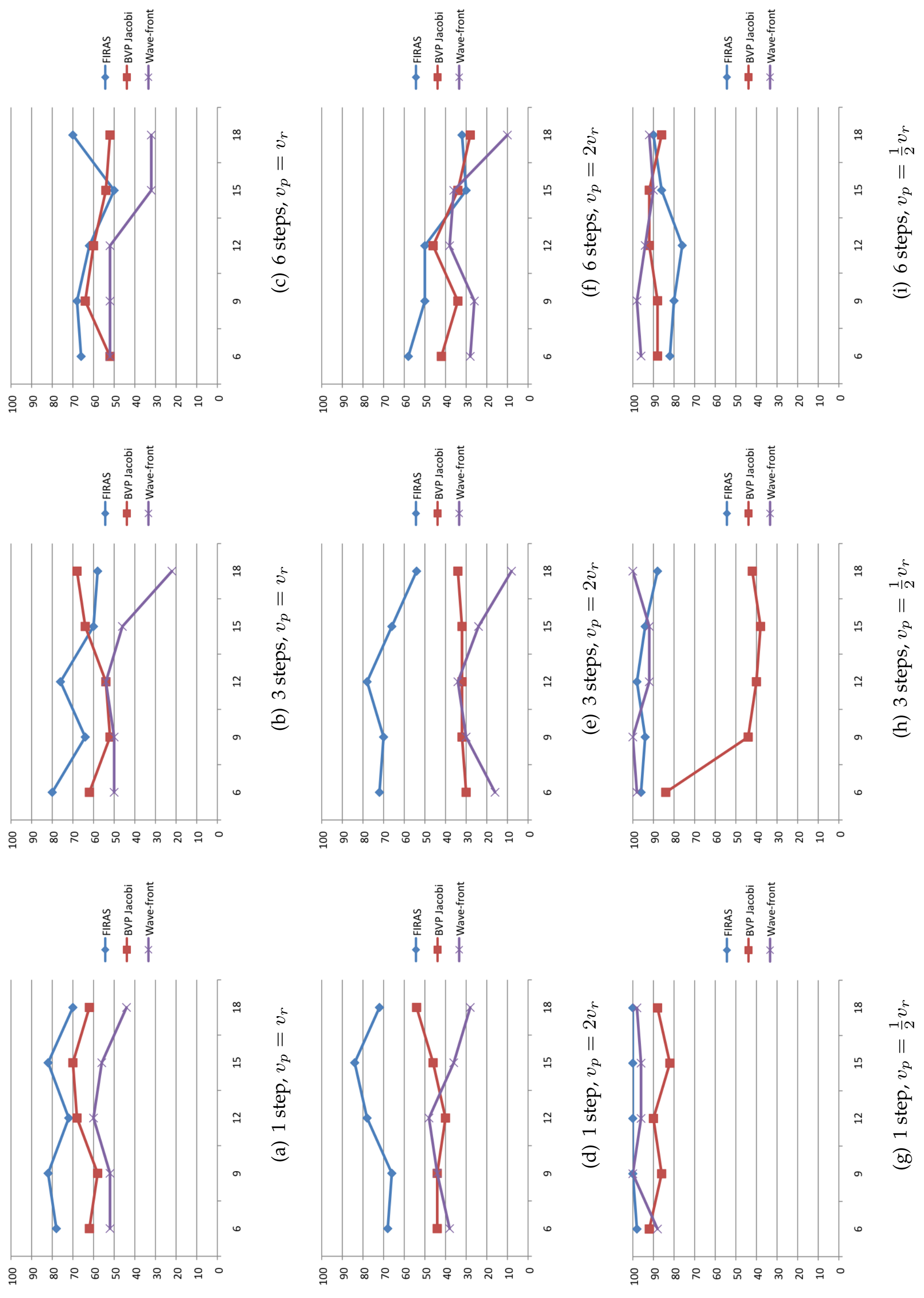
Comparing the results in Section 5.3, the following conclusions can be drawn:

- For fewer pursuers (1, 2 or 3), updating the sensing more frequently leads to better results. The same observation was made for the low resolution grids and slow pursuers, but in the case of higher resolution grids, increments in the performance due to frequent sensing has extended up to 3 pursuers, and all the speeds (fast and slow).
- For larger numbers of pursuers (more than 3) and regardless of their speed, updating the sensing less frequently has a similar effect or improves the results compared to frequent updates. The same observation was also made for the low resolution grids.

The best radius for computing the subgoal based paths is $r_{sg} = 6$ cells (equivalent to 2 in low resolution grids), for all numbers of slower pursuers, and the best potential functions: modified FIRAS and quadratic goal, and wave-front. This condition changes for faster pursuers, where the radius shifts to between 9 and 12 cells. It can be interpreted as a sign of the need for a straighter motion towards a location away from pursuers, to improve the possibilities to escape capture.

Statistics of the times of capture were also computed as in Subsection 5.3.2, for a maximum of 30 navigation steps. Figures 5.35 to 5.37 present some of the results in the form of box plots, for the different used potential functions. As previously, the results for 1, 3 and 10 pursuers are shown, for sensing updates of 3, 9 and 18 navigation steps. Other box plots for 2,3, 4 pursuers are shown in Appendix E.

The plots of the capture times measured in navigation steps confirm the dominance of the modified FIRAS and quadratic goal, combined with steepest descent, as the most successful potential functions. Exceptionally, wave-front potential functions, also combined with steepest descent, are the most successful in the case of 10 pursuers present in the environment. The average capture times for all the different potential functions become almost identical when the pursuers in the experiment are faster than the robot, for more than one pursuer.



(a) 1 step, $v_p = v_r$

(b) 3 steps, $v_p = v_r$

(c) 6 steps, $v_p = v_r$

(d) 1 step, $v_p = 2v_r$

(e) 3 steps, $v_p = 2v_r$

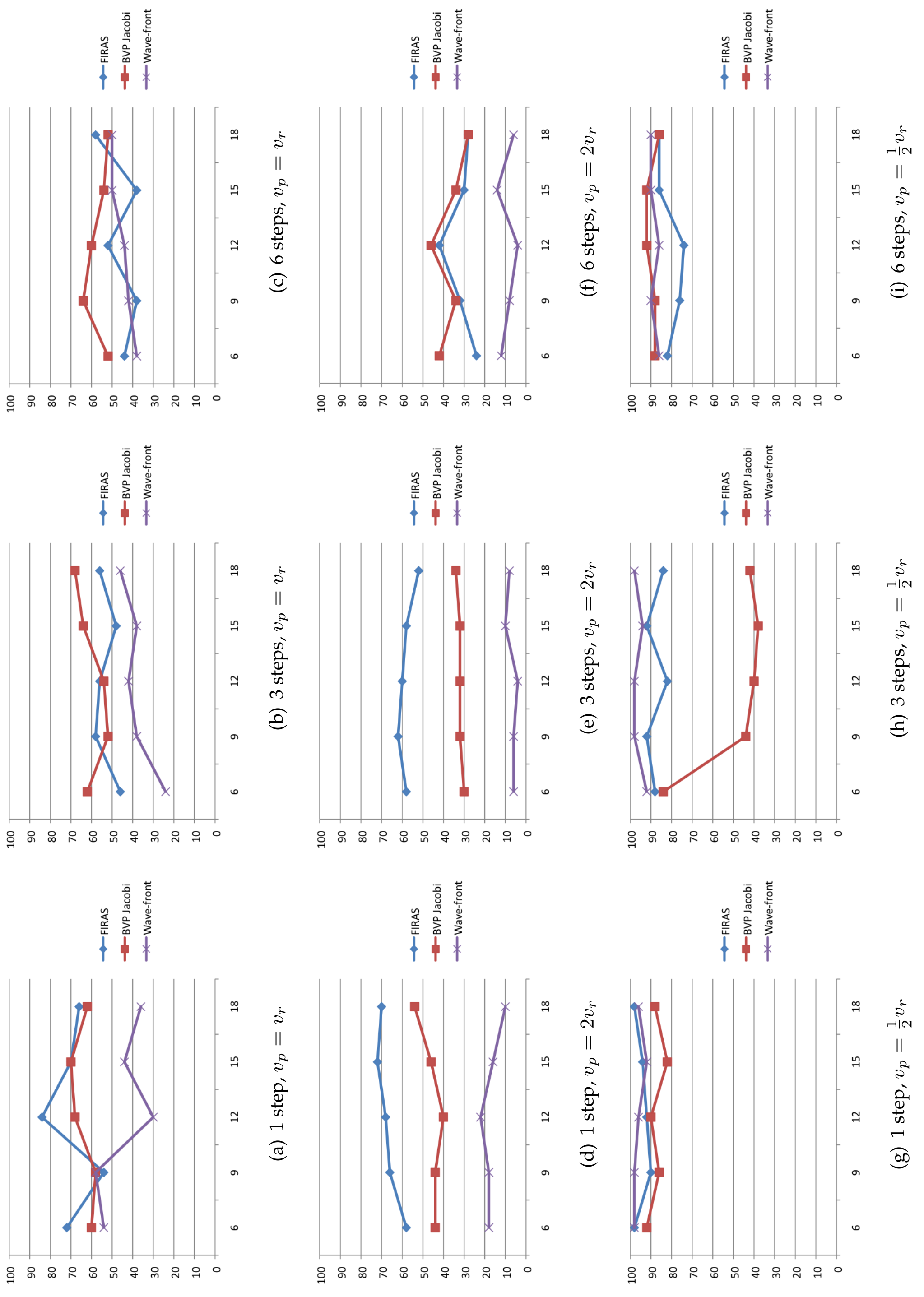
(f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$

(h) 3 steps, $v_p = \frac{1}{2}v_r$

(i) 6 steps, $v_p = \frac{1}{2}v_r$

Figure 5.29: Success of steepest descent paths for 1 pursuer and a higher resolution grid, for different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 1 step, $v_p = v_r$

(b) 3 steps, $v_p = v_r$

(c) 6 steps, $v_p = v_r$

(d) 1 step, $v_p = 2v_r$

(e) 3 steps, $v_p = 2v_r$

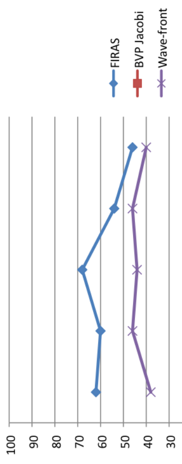
(f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$

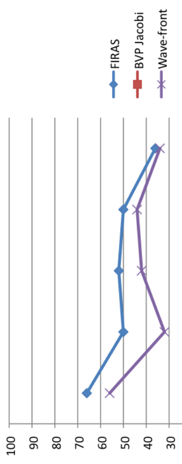
(h) 3 steps, $v_p = \frac{1}{2}v_r$

(i) 6 steps, $v_p = \frac{1}{2}v_r$

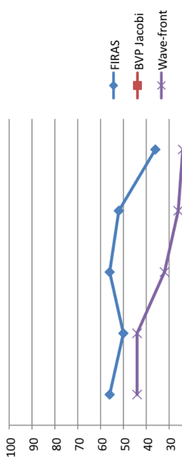
Figure 5.30: Success of subgoal based paths for 1 pursuer and a higher resolution grid, for different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



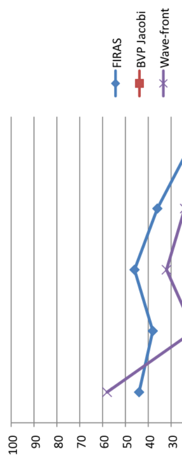
(a) 1 step, $v_p = v_r$



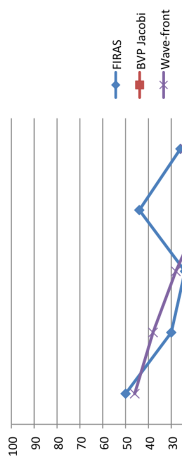
(b) 3 steps, $v_p = v_r$



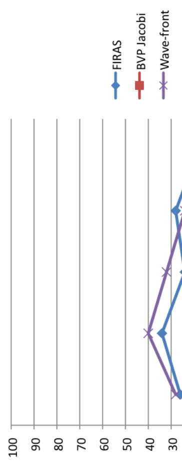
(c) 6 steps, $v_p = v_r$



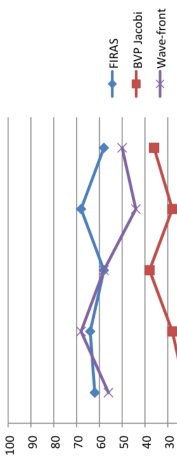
(d) 1 step, $v_p = 2v_r$



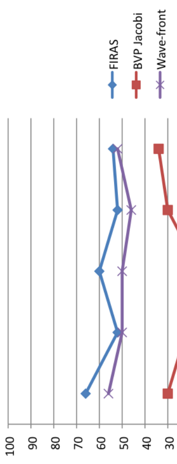
(e) 3 steps, $v_p = 2v_r$



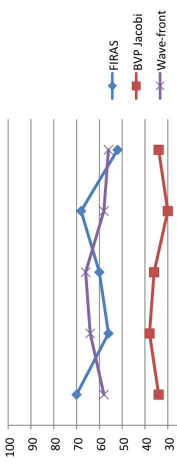
(f) 6 steps, $v_p = 2v_r$



(g) 1 step, $v_p = \frac{1}{2}v_r$

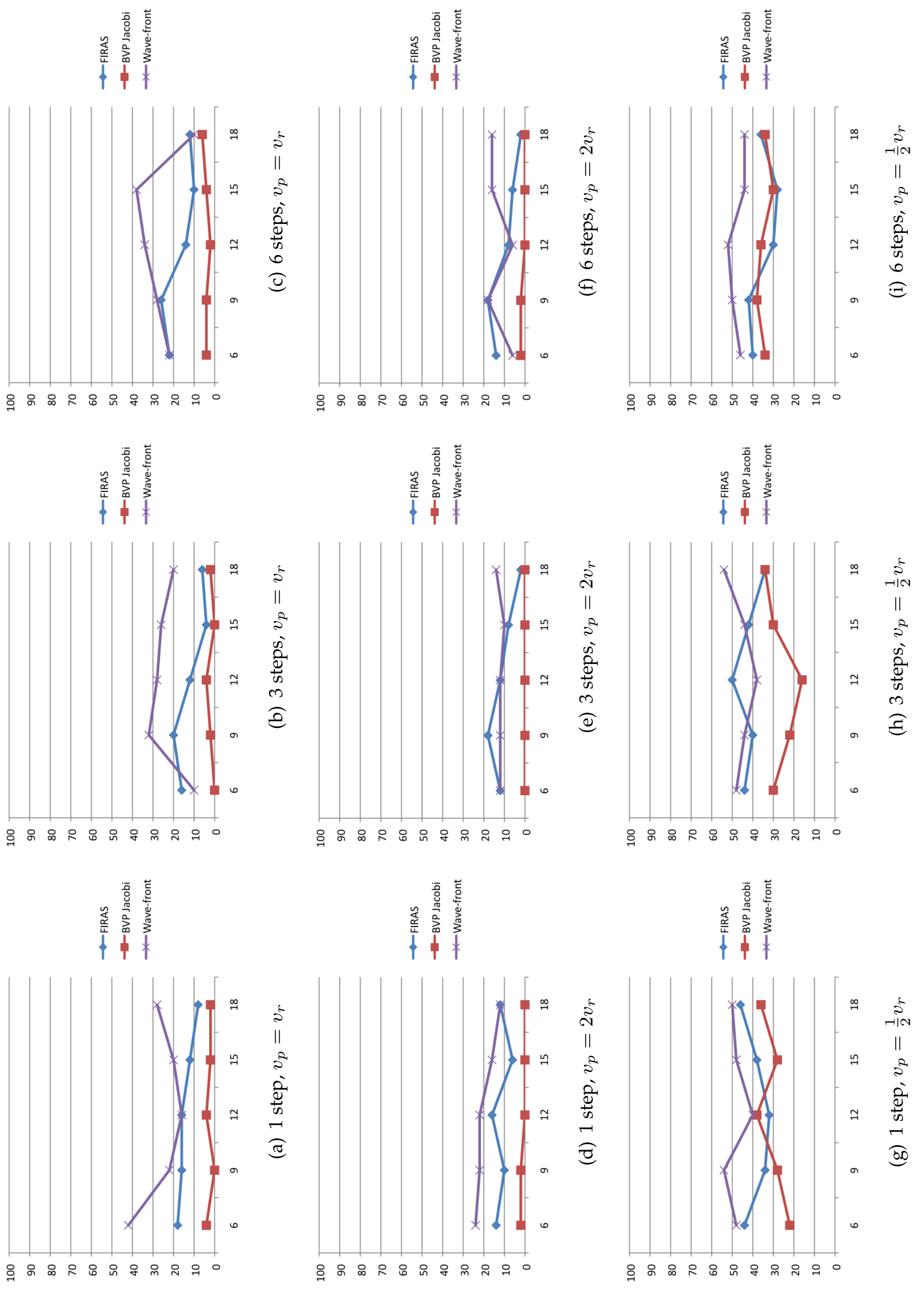


(h) 3 steps, $v_p = \frac{1}{2}v_r$



(i) 6 steps, $v_p = \frac{1}{2}v_r$

Figure 5.31: Success of steepest descent paths for 3 pursuers and a higher resolution grid, for different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.

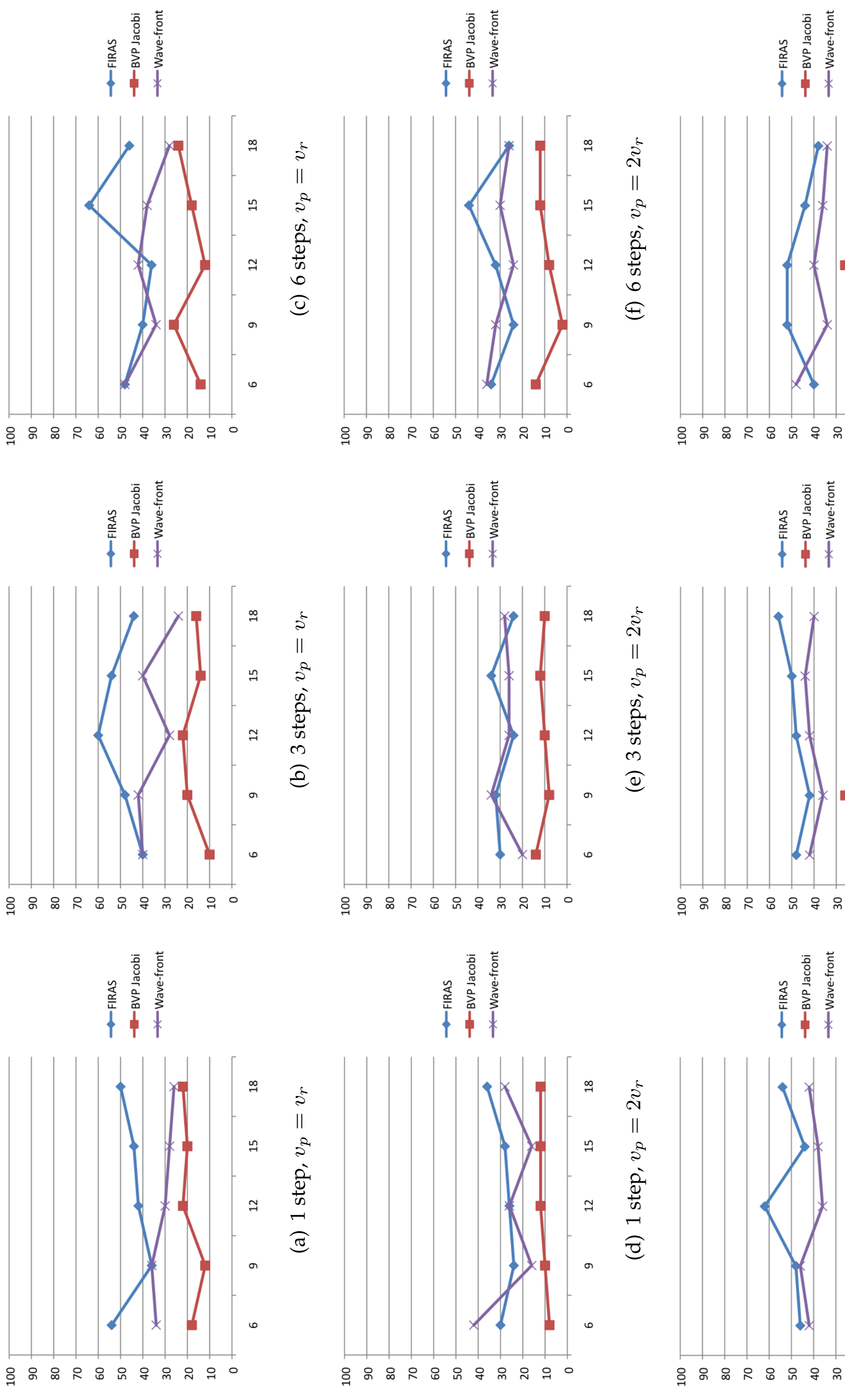


(a) 1 step, $v_p = v_r$ (b) 3 steps, $v_p = v_r$ (c) 6 steps, $v_p = v_r$

(d) 1 step, $v_p = 2v_r$ (e) 3 steps, $v_p = 2v_r$ (f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$ (h) 3 steps, $v_p = \frac{1}{2}v_r$ (i) 6 steps, $v_p = \frac{1}{2}v_r$

Figure 5.32: Success of subgoal based paths for 3 pursuers and a higher resolution grid, for different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 1 step, $v_p = v_r$

(b) 3 steps, $v_p = v_r$

(c) 6 steps, $v_p = v_r$

(d) 1 step, $v_p = 2v_r$

(e) 3 steps, $v_p = 2v_r$

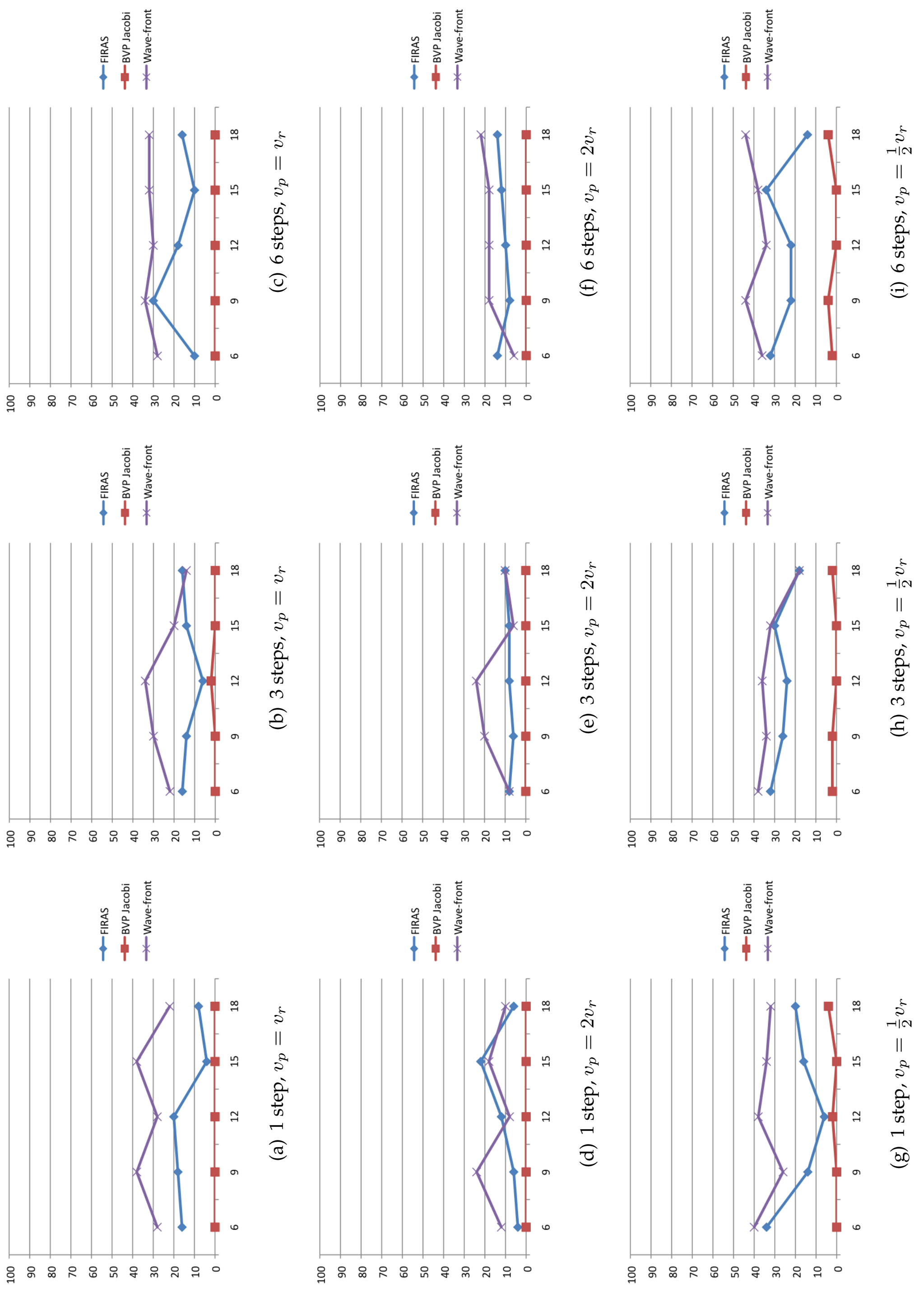
(f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$

(h) 3 steps, $v_p = \frac{1}{2}v_r$

(i) 6 steps, $v_p = \frac{1}{2}v_r$

Figure 5.33: Success of steepest descent paths for 10 pursuers and a higher resolution grid, for different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 1 step, $v_p = v_r$

(b) 3 steps, $v_p = v_r$

(c) 6 steps, $v_p = v_r$

(d) 1 step, $v_p = 2v_r$

(e) 3 steps, $v_p = 2v_r$

(f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$

(h) 3 steps, $v_p = \frac{1}{2}v_r$

(i) 6 steps, $v_p = \frac{1}{2}v_r$

Figure 5.34: Success of subgoal based paths for 10 pursuers and a higher resolution grid, for different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.

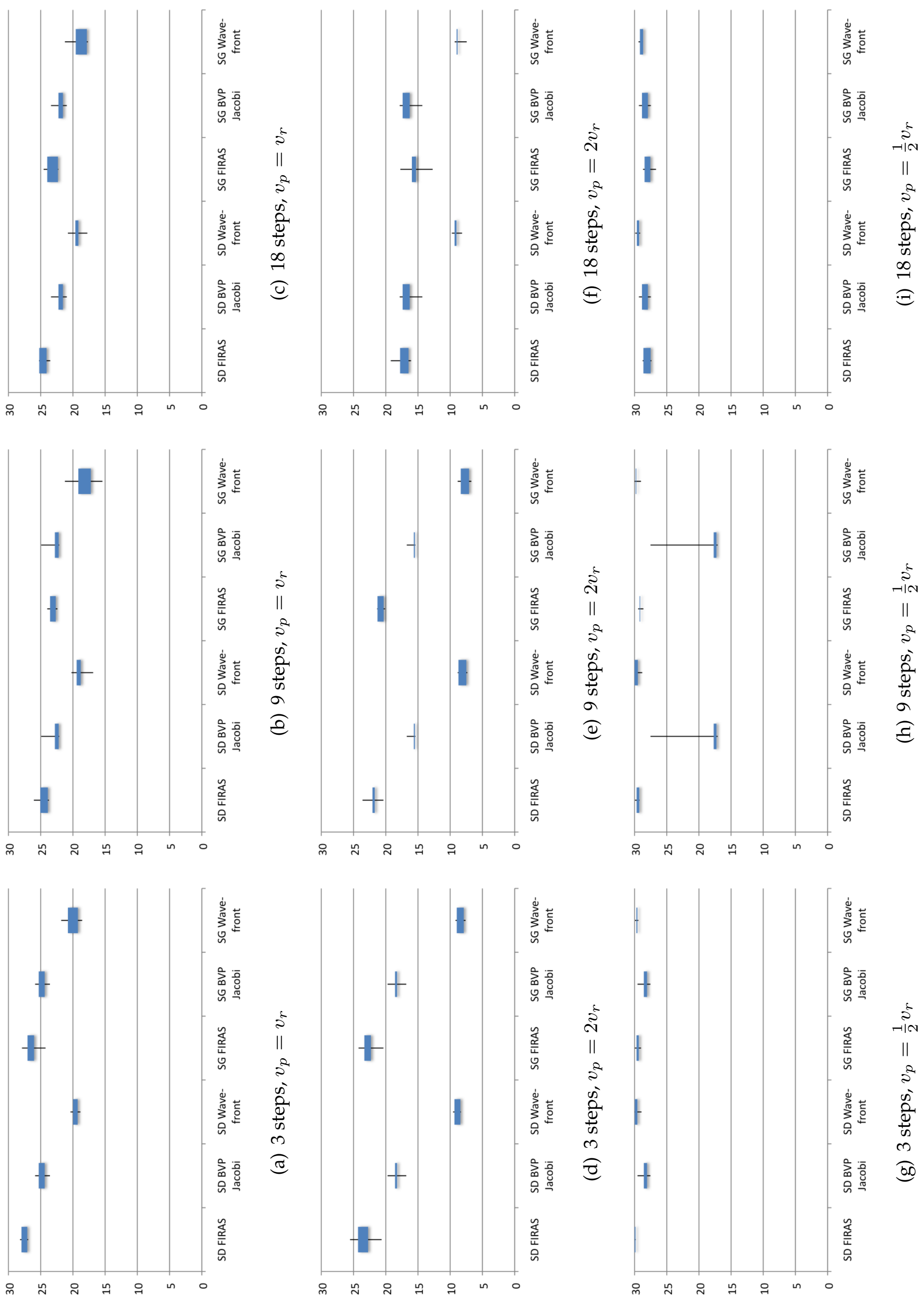
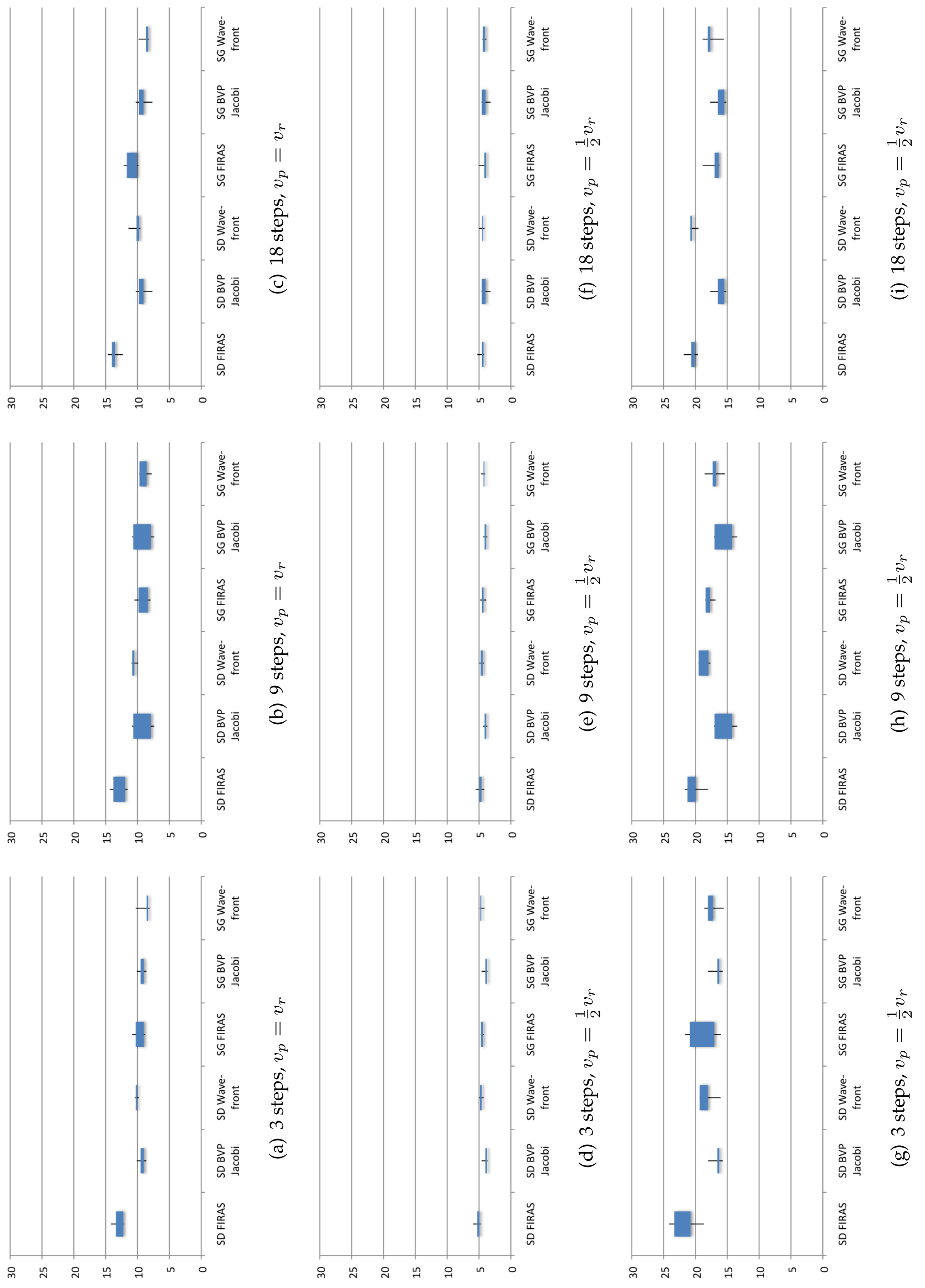


Figure 5.35: Capture times (in terms of navigation steps) for 1 pursuer, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 50 simulations.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = \frac{1}{2}v_r$

(e) 9 steps, $v_p = \frac{1}{2}v_r$

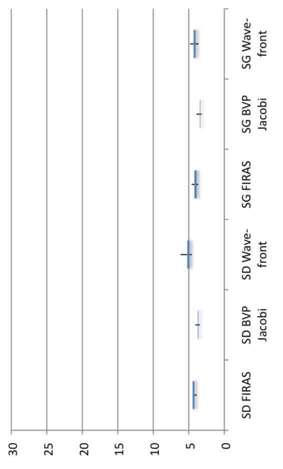
(f) 18 steps, $v_p = \frac{1}{2}v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

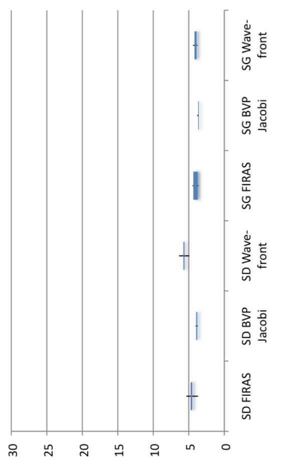
(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

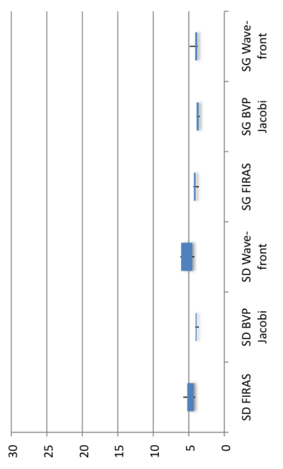
Figure 5.36: Capture times (in terms of navigation steps) for 3 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 50 simulations.



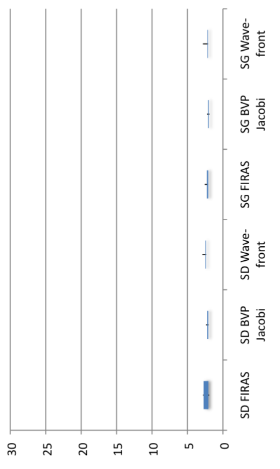
(a) 3 steps, $v_p = v_r$



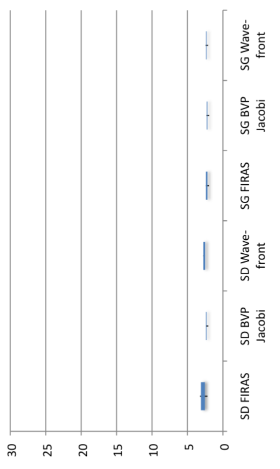
(b) 9 steps, $v_p = v_r$



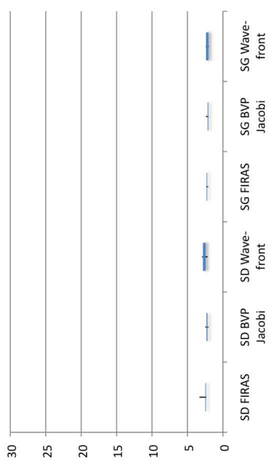
(c) 18 steps, $v_p = v_r$



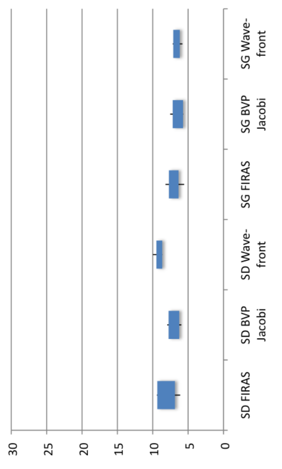
(d) 3 steps, $v_p = \frac{1}{2}v_r$



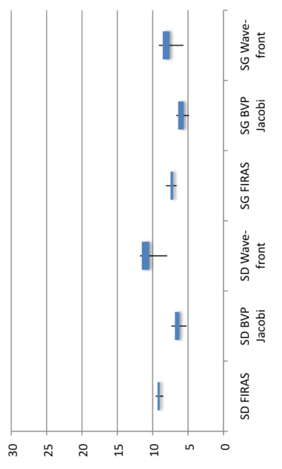
(e) 9 steps, $v_p = \frac{1}{2}v_r$



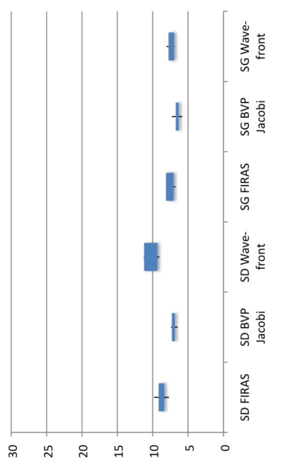
(f) 18 steps, $v_p = \frac{1}{2}v_r$



(g) 3 steps, $v_p = \frac{1}{2}v_r$



(h) 9 steps, $v_p = \frac{1}{2}v_r$



(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure 5.37: Capture times (in terms of navigation steps) for 10 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 50 simulations.

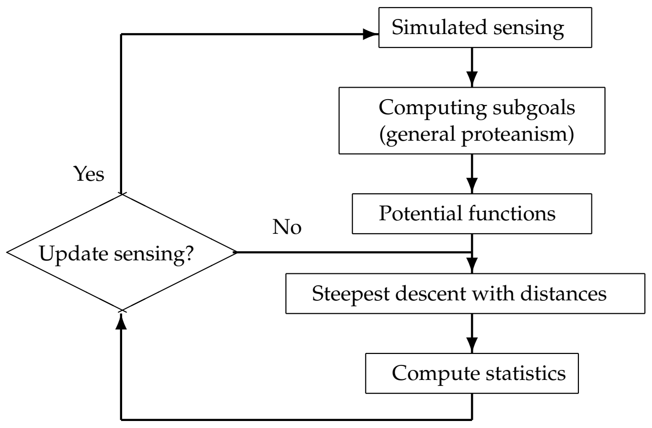


Figure 5.38: Flow diagram of the computations for the simulations to evaluate the success avoiding capture while pursuers move, for low resolution grids, and updating the sensing. Both paths, steepest descent only and subgoal based (protean) are computed independently in the simulations, but following the flow of processing shown in the diagram, as the consequences of the navigation strategies in terms of navigation and motion of the pursuers are different.

The results with the higher resolution grids confirm the consequences of the variations in the number of pursuers and their speed. Faster pursuers and in larger quantities capture the robot more frequently in the environments, due to the cluttering in combination with obstacles, and the coverage of larger portions of the free space. Consequently, in local environments with only one pursuer around, the possibility of escaping is much higher than with more pursuers.

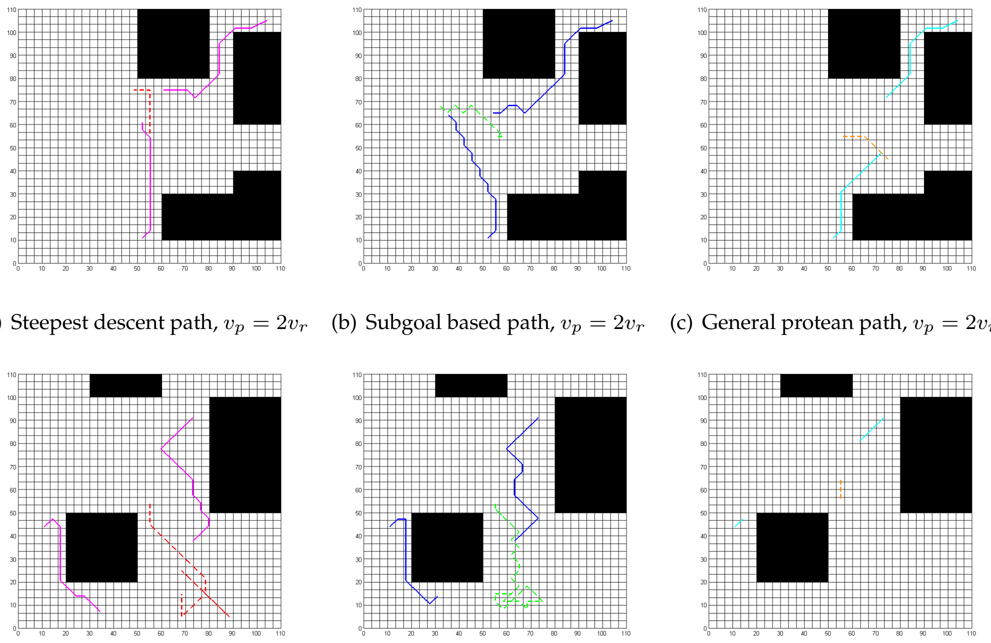
5.6 General proteanism, high resolution, sensing updates

The objective of this section is evaluating the performance of an entirely random constructed path (that would still avoid collisions against obstacles) through a more general computation of subgoals, mentioned in Section 4.4.5. Figure 5.38 presents the flow of the computations for the general protean path.

The success of general proteanism paths for different parameters is compared with steepest descent only paths and guided proteanism paths, the latter both towards a local goal that changes according to updates in the sensing. In the simulations of this section, the success is defined as avoiding capture before 30 navigation steps.

The parameters for the simulations have been defined in Table 5.2, for high resolution grids and sensing updates. Modified FIRAS and quadratic goal potential functions were used to compute the simulations in this section, as they offered the highest success in previous sections.

Subsection 5.6.1 presents some visual examples of computed paths, using the proposed steepest descent, guided proteanism and general proteanism approaches. Subsection 5.6.2 presents a statistical analysis of the success, through comparative graphs.



(a) Steepest descent path, $v_p = 2v_r$ (b) Subgoal based path, $v_p = 2v_r$ (c) General protean path, $v_p = 2v_r$

(d) Steepest descent path, $v_p = \frac{1}{2}v_r$ (e) Subgoal based path, $v_p = \frac{1}{2}v_r$ (f) General protean path, $v_p = \frac{1}{2}v_r$

Figure 5.39: Concentrated paths (protean general and guided, steepest descent) simulating the navigation of the robot and the pursuers, and updating the sensing after 3 navigation steps, for a higher resolution grid. Steepest descent paths in red and corresponding pursuers in magenta, subgoal based paths in green and corresponding pursuers in blue, and general protean paths in orange and corresponding pursuers in cyan; obstacles in black.

5.6.1 Visual examples of computed paths

This subsection presents some examples of computed paths for 2 pursuers with $v_p = v_r$, sensing updates every 3 navigation steps, $r_{sg} = 6$ for general and guided proteanism and modified FIRAS and quadratic goal potential functions. Other parameters were used as indicated in Table 5.2. Figure 5.39 shows the examples, and more have been added in Appendix C.

5.6.2 Success of computed paths

This subsection presents a comparison of the performance of a general protean path and the other two proposals, steepest descent and guided proteanism towards a local goal. The success is computed only on base of the avoidance of capture for a maximum of 30 navigation steps, for environments with different numbers of pursuers and their velocities. Other parameters for the simulations were selected according to Table 5.2 for high resolution grids with sensing updates. Modified FIRAS and quadratic goal were used to compute the paths.

Significative statistics on the success of the paths are presented in Figure 5.40 for 1 pursuer, Figure 5.41 for 3 pursuers, and Figure 5.42 for 10 pursuers, respectively, and for sensing updates of 3, 9 and18 navigation steps. Results for 2, 4 and 5 pursuers are shown in Appendix E. The legends in the plots are: Steepest descent, Subgoals and General protean, for the three respective kinds of paths.

The measurement of the success in the series of simulations shown in this section allowed the analysis of the absolute performance of the three different kinds of paths, protean guided, protean general and steepest descent only, to find the most successful implementation against capture by pursuers. The avoidance of capture for 30 navigation steps with each different path was used as the measure of success, without including a comparison of the performance between each other as previously done in Sections 5.3 and 5.5.

Only modified FIRAS and quadratic goal potential functions were evaluated as they demonstrated to be the best option according to the relative success measured in the previous sections of simulations. Steepest descent based paths resulted in better avoidance of capture for the different numbers of pursuers and their velocities, compared to any kind of proteanism, guided or general. Guided paths, both protean and steepest descent, performed better than general proteanism (or random navigation).

When only one pursuer is present in the environment, the avoidance of capture is more likely, even for a fast pursuer. For more than one pursuer, the success avoiding faster or equivalent speed pursuers reaches the lowest values. This particular feature is closely related to the reported analysis of capture with multiple pursuers at different speeds, particularly popular in visibility-based or graph theory solutions to pursuit-evasion games. A small local environment with more and faster pursuers decreases dramatically the possibilities to escape capture, even when using proteanism. On the other hand, when dealing with slower pursuers and despite their number, more possibilities to avoid capture are available, noted in the percentages in the corresponding figures.

Statistics of the times of capture were also computed. Figures 5.43 to 5.45 present some of the results in the form of box plots, for the same numbers of pursuers and updates. As previously, the results for 1,3,10 pursuers and sensing updates of 3,9,18 navigation steps are presented. Other box plots for 2,3, 4 pursuers are shown in Appendix E.

The computed times of capture are the same as the ones in Section 5.3, for the steepest descent only and guided proteanism (subgoals) based paths. General proteanism did not improve the average time of capture for any number of pursuers or the differences on their speeds. Nevertheless, it is possible to appreciate that guided subgoals improve or show an equivalent performance to steepest descent only paths when the pursuers are faster (more notorious in the case of 1 pursuer).

The steepest descent only paths are ideal to deal with pursuers of equivalent speed or slower, whereas guided subgoal based paths offer a reasonable solution when the pursuers are faster. Although for more than 1 pursuer, if the robot is fast enough, its chances of survival improve dramatically. A randomised navigation strategy without a clear goal does not exactly provide a better solution than one that mixes a dynamic local navigation goal and some randomness in the deliberation of a path.

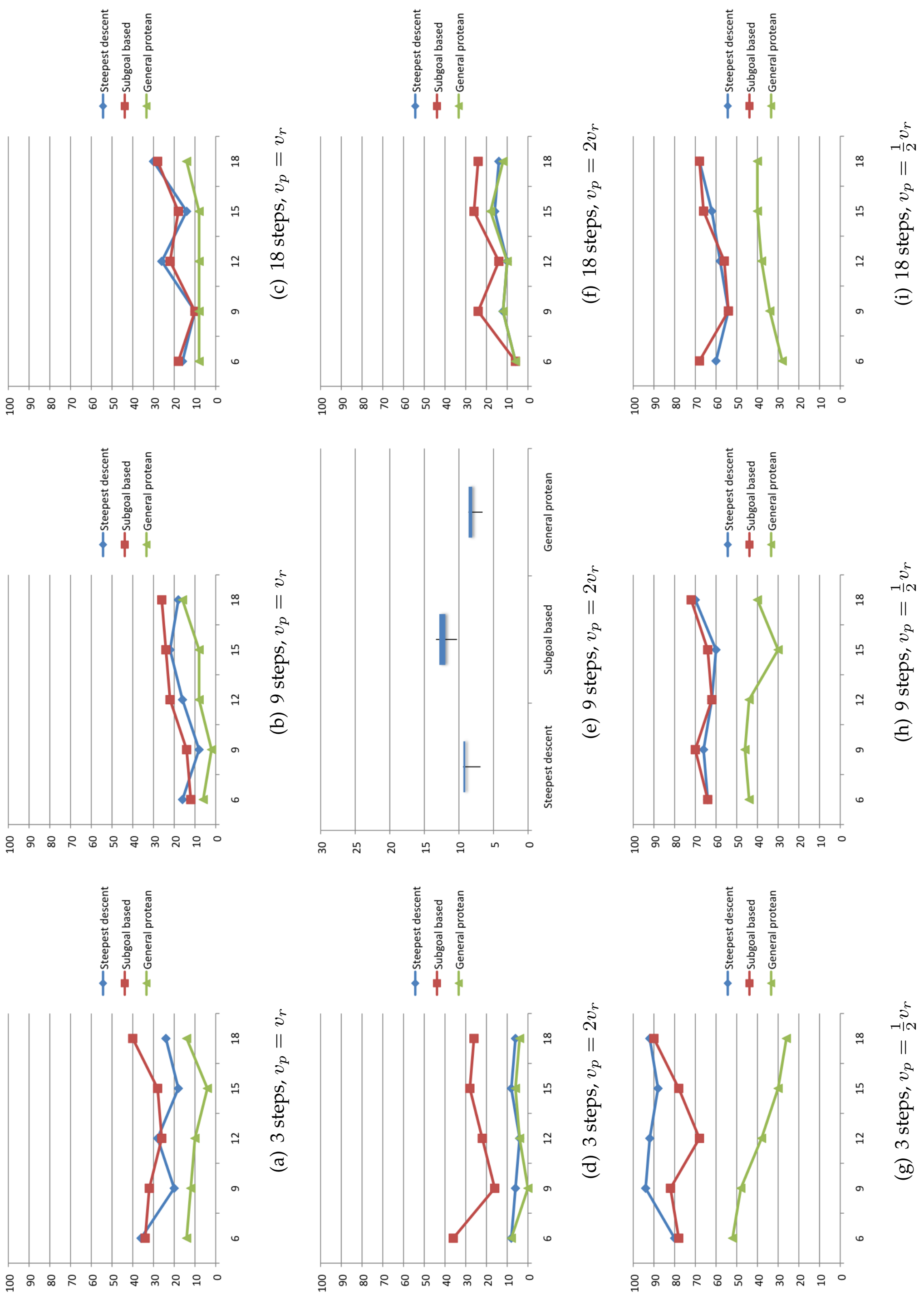
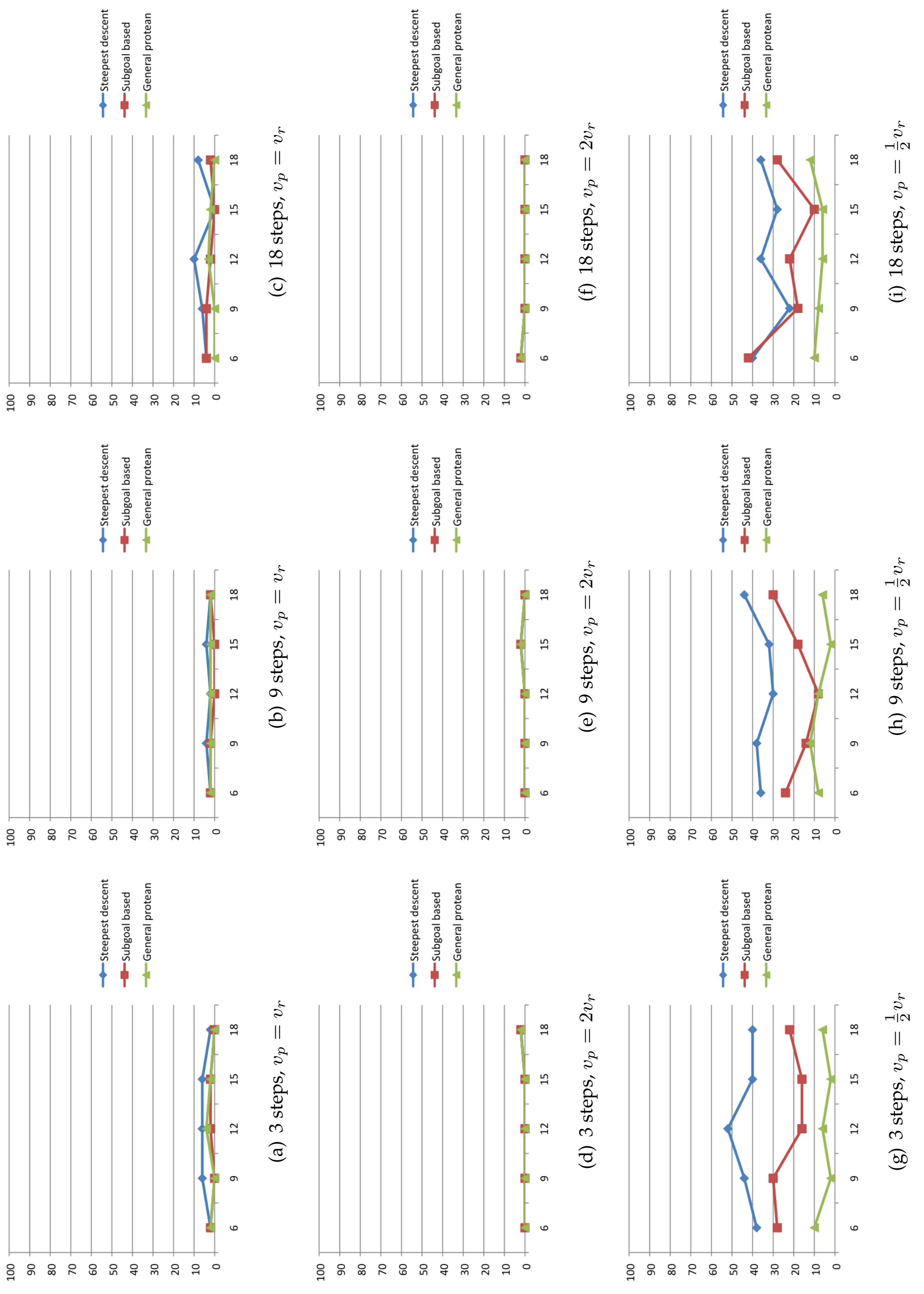


Figure 5.40: Success of the paths for 1 pursuer and a higher resolution grid, for different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure 5.41: Success of the paths for 3 pursuers and a higher resolution grid, for different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.

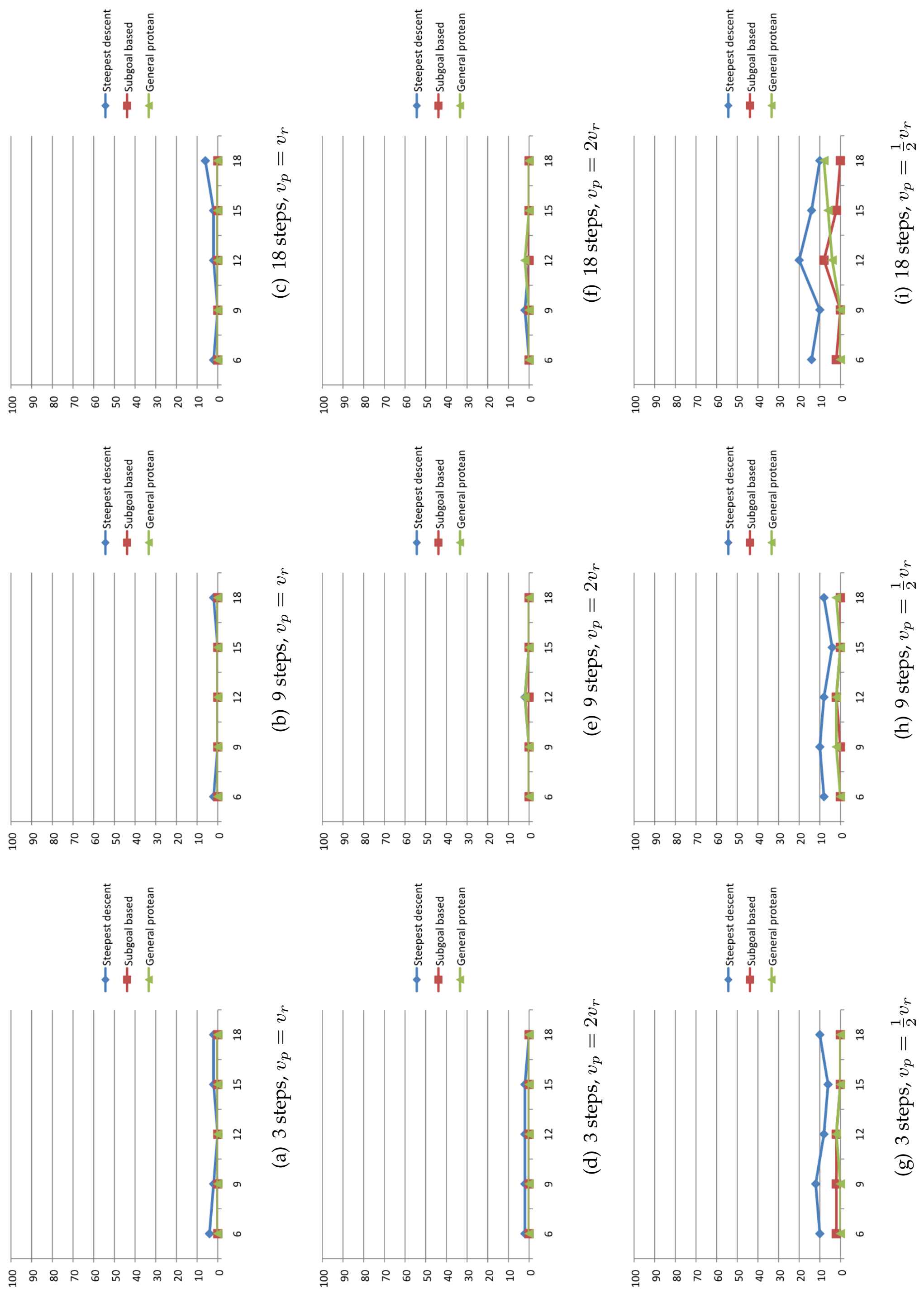
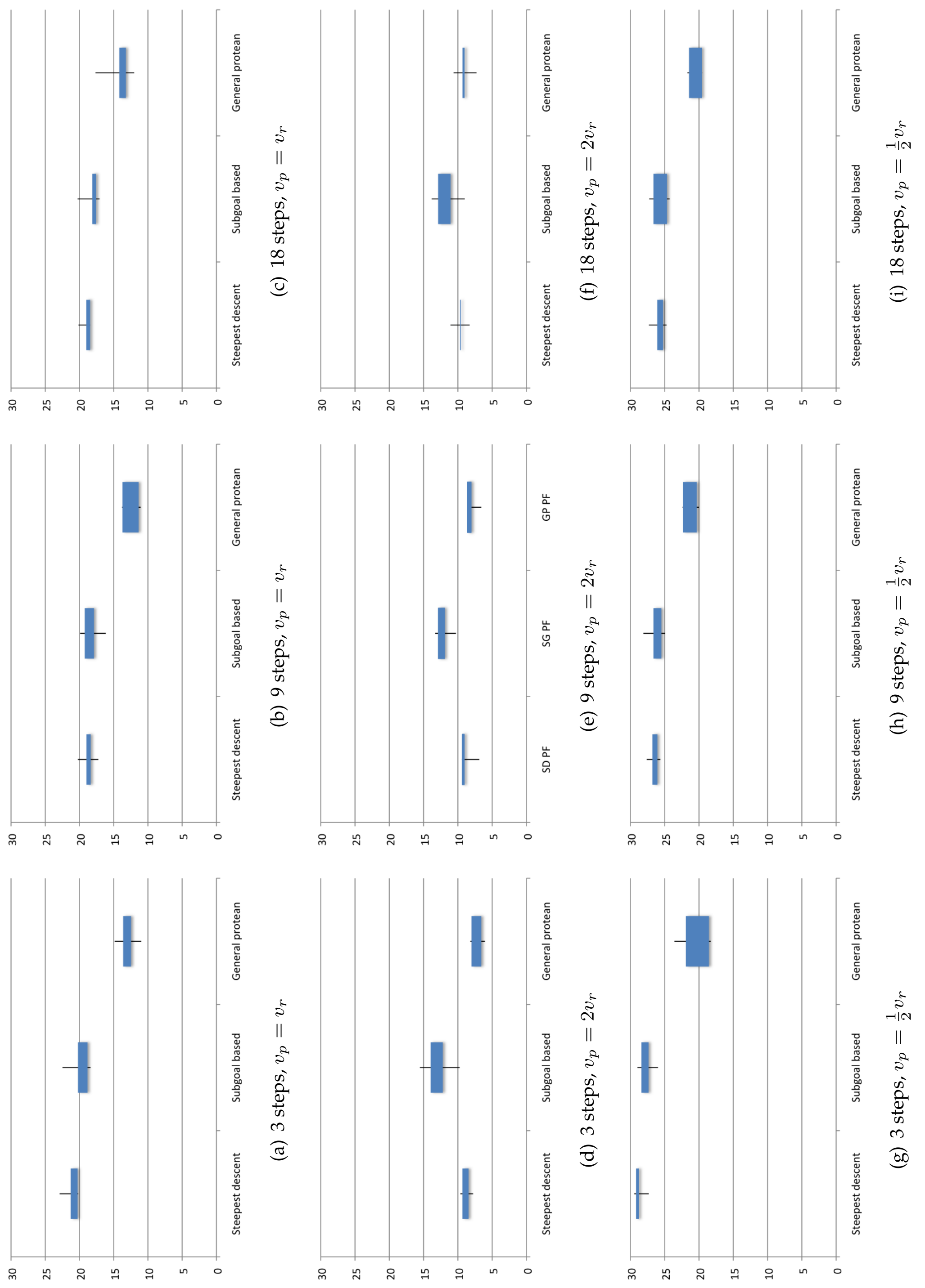


Figure 5.42: Success of the paths for 10 pursuers with different velocities in a higher resolution grid, for different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

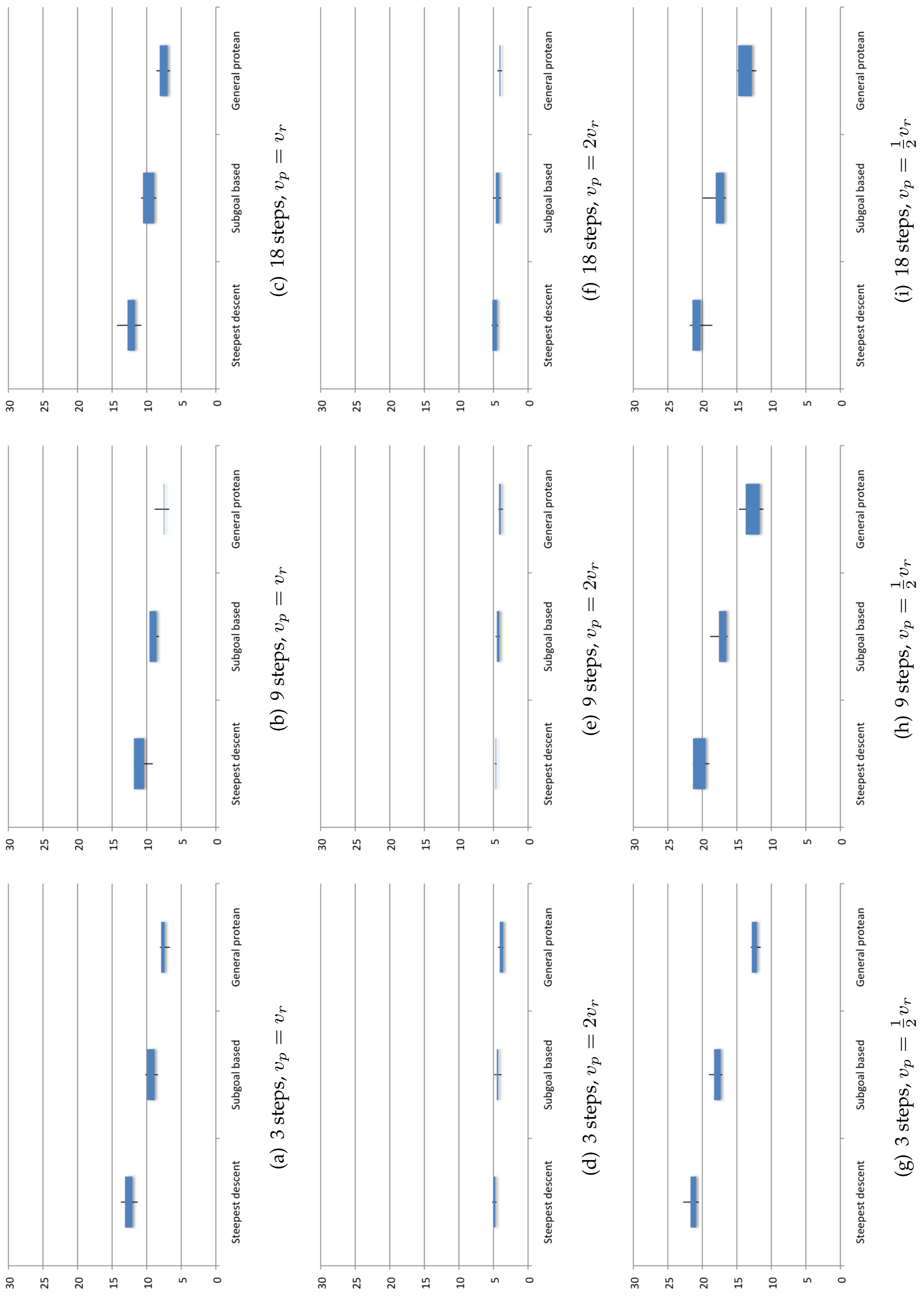
(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure 5.43: Capture times (in terms of navigation steps) for 1 pursuer and a higher resolution grid, for different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 50 simulations.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

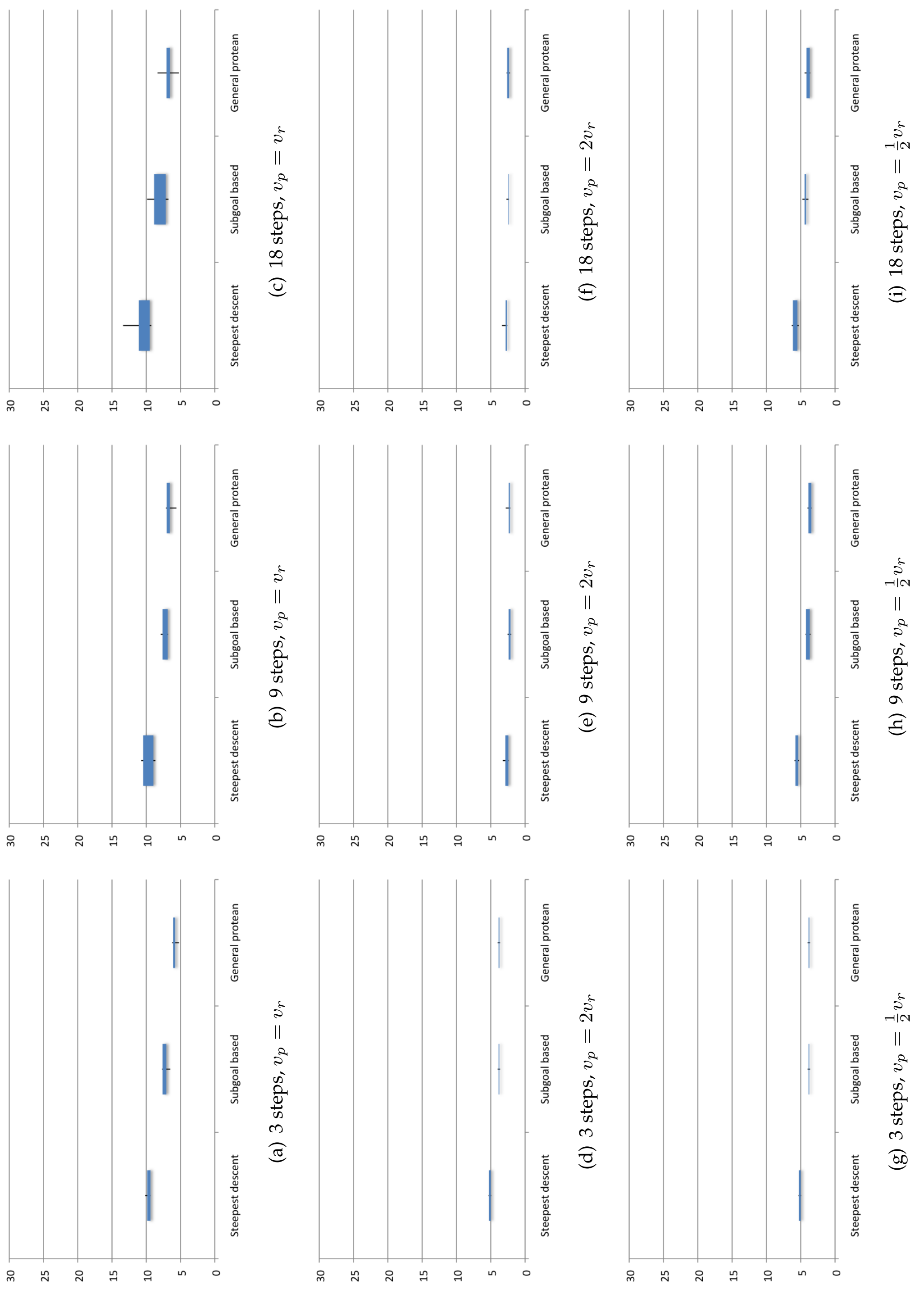
(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure 5.44: Capture times (in terms of navigation steps) for 3 pursuers and a higher resolution grid, for different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 50 simulations.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure 5.45: Capture times (in terms of navigation steps) for 10 pursuers and a higher resolution grid, for different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 50 simulations.

5.7 Discussion

The restriction in the amount of information from the environment due to the limits in sensing in a real robot means that the analysis of the environment needs to be updated to collect the new information as the robot moves. Therefore, an iterative process of the analysis of the environment and computation of a path is necessary. The grid occupancy representation used in this thesis for the proposed analysis of the environment has some disadvantages if the observed areas are quite reduced or closely blocked, and most of the space adjacent to the robot or behind current obstacles is unknown. Also, the use of a discretisation in the form of a grid restricts the computation of continuous paths to discrete, but this problem can be solved by techniques like smoothing or interpolation.

The randomly generated environments used for the simulations containing several pursuers and obstacles (cluttered environments) are not a realistic representation of many tasks of navigation for robots, but an exaggeration or worst-case scenario. The presence of numerous elements restricts the free space, and consequently the viability of a path that guarantees the escape from pursuers. Nevertheless, the generated environments with different numbers of pursuers are useful to evaluate the performances of the proposals. More simulations with complex, more realistic and wider environments are desirable to complement the evaluation presented in this chapter.

The paths generated through steepest descent and proteanism (subgoals) had expected characteristics. Both kinds of paths became more different when closer subgoals (smaller radius) were introduced in the protean paths, causing the visible zig-zag deviations or randomness. The differences in the paths were analysed by computing the DFD and the RMSE as analytic measures.

Subgoal based paths are better against the problem of local minima for static environments, but steepest descent functions are much better when the environment sensing is updated as the robot navigates and the elements in the environment change. This means that local minima problems become trivial when constant updates of the sensing and path planning are performed, and the changes in the path lead to the avoidance of a straight path by responding to the motion of the pursuers. If the environment was completely known in advance, a fixed path could be planned from a location to another, and then, the subgoals could help to avoid local minima problems due to the potential functions representation, as other authors have proposed (Bell, 2005). Nevertheless, the use of the subgoals in such a fixed path would also lead to a navigation similar to a bang-bang steering control that oscillates along the path, when the subgoals are computed and commuted closely enough. The optimal radius for computing the subgoals depends on the physical characteristics of the robot itself (particularly the size and the steering capabilities), as shown when changing the resolution of the grid, where the subgoals should not be too close to the current location of the robot, nor as far as the limits of the environment (otherwise it becomes an almost linear motion towards outside of the local environment).

Other solutions to the problem of local minima encountered in the paths for static environments would involve using different potential functions to the ones proposed in the designs, like potential vector fields. They would allow incorporating rotational motions around obstacles to avoid collisions, and they would facilitate the computation of a path by substituting the search method with the physics of following the flow in the field.

Finding when it is necessary to perform the updates for sensing depends on the knowledge of the environment and the capabilities of the robot: if the environment is known (e.g. from mapping), and models of the pursuers are available, then sensing updates are not that necessary to compute paths for navigation and escape. On the other hand, if the environment is entirely unknown, sensing is needed more frequently to guarantee collision avoidance and feedback about the pursuers responses. Also, if the pursuers can be modelled, from information, and the rest of the environment does not change radically in time, the sensing can become more sparse in time.

According to the results presented throughout the chapter, the best combinations of parameters for the proposed designs are:

- For few pursuers in the local sensed environment, a continuous update of the sensing, interpreted as a state of alert, is necessary to detect the pursuer as soon as possible and start an action to prevent capture. Steepest descent only computed paths show the best performance in general over the paths computed based on subgoals for high and low resolution grids, for both faster and slower pursuers.
- For many pursuers in close proximity, updates are not as necessary in a continuous manner to find any threat in the environment for low resolution grids representing the environment, as many pursuers are detected from the first sensing. The chances to avoid capture at all are much reduced in this kind of situation, but proteanism has proved to be useful to delay the capture for at least a couple of steps for low resolution grids regardless of the speed of the pursuers.

Other important results are the following:

- For low resolution grids, the modified FIRAS and quadratic goal and the wave-front potential functions perform well reaching the local goal (particularly if using subgoals rather than only steepest descent).
- The solutions to harmonic equations (Laplace and Laplace with perturbation), through a boundary value problem and Dirichlet conditions, present problems. The potential values are distributed in flat surfaces along the free space in just a couple of iterations of the numerical methods, and although there is only one minimum located in the goal, the flat patches cause a problem with the steepest descent search method. The resulting paths lead towards the edges of the grid, sometimes away from pursuers, but not towards a local goal.
- An improvement in the performance of the BVP solutions for the Laplace and Laplace with perturbation equations with iterative numerical methods is present when increasing the resolution of the grid.
- Modified FIRAS and quadratic goal potential functions, followed by the wave-front function, wre found to be the best when interacting with pursuers and updating the

environment sensing. Both functions are easy to compute for grids, and they do not present the problems observed with other methods in the boundaries of the local environment that is being processed.

- In general, higher resolution grids (in this case artificially modified for the simulations) produce an increase in the relative success of the paths compared with low resolution grids, as the robots are able to navigate for equivalent distances or for longer.

Avoiding capture entirely is not possible when faster and many pursuers are in close proximity. The results on absolute success in the last section of simulations confirm that. Visibility based analysis of pursuit-evasion games have reported that evaders would require a velocity of at least $1 + \frac{5}{\frac{n_p}{4} - 3}$ the velocity of the n_p pursuers, to avoid capture by multiple pursuers in an obstacle free environment (Klein and Suni, 2010). Thus, the only options that the robot has to avoid capture for longer are to use obstacles and any other trick to avoid being too predictable.

The model used for the pursuers in the simulations in this chapter is a simple version of greedy navigation. More complex behaviours incorporating communication and cooperation, and interception capabilities need to be explored in order to validate the performance of proposed designs.

5.8 Concluding remarks

Five groups of simulations to assess the proposed solutions to the problem of navigation and escape from pursuers, described in Chapter 4, were presented. The effect of variations of parameters and characteristics of the environment analysis and path planning with potential functions served to evaluate the performance of the generated paths, for different randomly generated environments. A summary of the values and variations used in the simulations is presented, followed by the simulations.

The first two groups of simulations evaluated the performance of the analysis of the environment and path planning for a low resolution grid representing the elements (pursuers, obstacles and free space). Visual examples of computed paths illustrate the computation of the paths. An analysis without updating the environment allowed verifying that the planned paths lead to a desired location in the environment, according to the mapping of the environment into the different potential functions. Then, an analysis with sensing updates (simulated) is performed, along with simulated pursuers of different velocities, to evaluate the computed paths in a more realistic setting.

The subsequent two groups of simulations evaluate the same parameters as the previous two, but for a higher resolution grid. The values of the parameters are adjusted in terms of the new resolution of the grid, to compare the results from both resolutions. More visual examples of the paths are presented.

In the final group of simulations, a more general protean approach is compared to the guided planning proposals, also in a higher resolution grid, with simulated pursuers, and the same parameter variations. The analysis presented throughout the chapter allow selecting the best combinations of potential functions and other computation parameters according to the most relevant characteristics of the environment: the pursuers.

The discussion of the results highlights the main findings throughout the simulations, with respect to the analysis of the environment and its limitations, the performance of the steepest descent and subgoal based paths, and the limitations of the simulations. A summary of the best alternatives and their parameters, from the ones proposed in Chapter 4, for the computation of paths as a solution to the problem of navigation and escape is included.

The main results include: modified FIRAS and quadratic goal as the best of all potential functions, lower radius of computation of subgoals to achieve the highest success when using protean paths, steepest descent paths performing better than protean paths except in cluttered environments, higher resolution grids increasing the success of the computed paths and the unavoidable capture when the environment is cluttered by pursuers. Local minima problems are present in complex geometries and discrete environments, thus the need to find alternatives that deal with all their different varieties. More simulations with complex and realistic environments (including the behaviour of pursuers) are needed in order to continue the validation of the proposed designs in this thesis.

The next chapter presents a comprehensive literature review on pursuit-evasion games and reinforcement learning, with main focus on proposed methods to compute strategies for evasion from unknown pursuers, and for mobile robotics. Different kinds of solutions to pursuit-evasion games are reviewed, along with the main characteristics and considerations for the implementation of reinforcement algorithms.

Chapter 6

Pursuit-evasion games and reinforcement learning

The navigation and escape problem has been analysed from two different perspectives in this thesis:

1. As a path planning problem, where an optimal trajectory of navigation is found according to the characteristics of the environment, different kinds of constraints, and a possible goal.
2. As a pursuit-evasion game, where the opponents are the pursuers (traditionally the other way around), and a navigation strategy is computed by optimising the payoff or value of the game, according to the state of the environment and the respective actions of the pursuers (from sensing).

This chapter reviews the concept of pursuit-evasion games from a game theory perspective, the main groups of proposed solutions to pursuit-evasion games, and the concepts and main algorithms from the area of reinforcement learning.

Pursuit-evasion games are introduced in Section 6.1, including their definition and common characteristics from reviewed literature. Pursuit-evasion games provide a different side to the solution of the problem of escape from pursuers, although research normally focuses on designing strategies for the pursuers, not for the evaders. Pursuit-evasion games have been solved using numerous techniques from artificial intelligence, including reinforcement learning, and multi-agent systems cooperation mechanisms. This section also mentions some of the main variations of pursuit-evasion games according to the characteristics of the players in the game, followed by some examples of applications to the solution of real life problems.

Section 6.2 presents a summary of different methodologies that have been used to solve the pursuit-evasion games and the use of reinforcement learning algorithms. Also, particular interest has been paid to finding examples of games where the evaders display an intelligent behaviour, different from a simple reactive navigation, or a randomised movement without a goal, as a source of inspiration to the development of strategies for navigation and escape from pursuers. Subsection 6.2.1 summarises the solutions to pursuit-evasion games as differential games. Subsection 6.2.2 mentions other proposed forms to compute controllers for pursuit. Subsection 6.2.3 presents some examples of co-evolved strategies for pursuit and evasion using genetic algorithms. Subsection 6.2.4 explores the use of visibility-based analysis (also called graph search methods) to represent and compute solutions of pursuit-evasion games. Subsection 6.2.5 presents some examples of the use of probabilistic representation of the environment due to incomplete information and the consequent optimal search computation of strategies for pursuit. Subsection 6.2.6 mentions some of the main proposals to solve multi-player pursuit-evasion games.

Subsequently, Section 6.3 presents a brief overview of reinforcement learning, starting with its definition. This section includes some examples of the use of reinforcement learning for autonomous navigation, including the extension of the navigation strategies to multi-robot systems from multi-agent reinforcement learning methods. Section 6.4 briefly summarises the main algorithms used for reinforcement learning: Monte Carlo, temporal difference (including Q-learning, Sarsa and the actor-critic architecture), temporal difference with eligibility traces and function approximation for continuous and large data action-state space sets. Subsection 6.4.4 presents the options to select the parameters used within reinforcement learning, including popular action selection mechanisms that promote exploration and exploitation in the search for computed optimal strategies or policies, and the use of dynamic parameters to vary the speed of the learning. Subsection 6.4.5 mentions some applications of reinforcement learning algorithms to the computation of navigation and evasion strategies.

Finally, Section 6.5 discusses the solutions to pursuit-evasion games proposed in literature, towards the design of strategies for navigation and escape from pursuers, and the use of reinforcement learning for path planning as an alternative to the potential functions used in Chapters 4 and 5. Section 6.6 presents the concluding remarks of the chapter.

6.1 Pursuit-evasion games

Game theory is an area of knowledge that studies definitions and solutions of games, which are interactions between different players. The applications of game theory include politics, economics, psychology, operations research, communications, manufacture, planning, scheduling, computer science, statistics, evolutionary biology and ecology. Many kinds of games have been researched in detail since the 1950s, to model and find a solution to real life problems. Pursuit-evasion games are one of the popular testbeds in distributed artificial intelligence for the evaluation of cooperation strategies (Isaacs, 1965).

In pursuit-evasion games, the players are pursuers and evaders. Considering a finite two-dimensional environment, $\mathcal{X}$, divided into square cells, the main elements of pursuit-evasion games are (Isaacs, 1965; Hespanha et al., 2000; Vidal et al., 2002):

- A set of cells, $k_o \subset \mathcal{X}$, containing a number of n_o fixed obstacles.

- A set of cells, $k_p \subset \mathcal{X}$, occupied by the n_p pursuers.
- A set of cells, $k_e \subset \mathcal{X}$, occupied by the n_e evaders.

Pursuers and evaders are allowed to move to cells in $\mathcal{X}$ in which there is no other pursuer, evader or obstacle. Each pursuer collects information about $\mathcal{X}$ at discrete time instants. The collected information $y(t)$ is a triple $< p(t), e(t), o(t) >$, where $p(t)$ denotes the measured positions of the pursuers (as cells), $e(t)$ is a set of cells where evaders are detected and $o(t)$ is the set of cells of detected obstacles. $\mathcal{Y}$ is defined as the sequence of collected information $\{y(1), ..., y(N)\}$ taken up to time N.

Considering $x_p(t) \in p(t)$ and $x_e(t) \in e(t)$ as the estimated positions of a pursuer p and evader e at time t, respectively, evader e is captured by pursuer p at time t if

$$d(x_p(t), x_e(t) \leq d_m \tag{6.1}$$

where $d(\cdot, \cdot)$ is a metric in $\mathcal{X}$ and d_m is a pre-specified capture distance. Captured evaders are removed from the game. The capture time of all evaders is defined as $t^* = \max t_e^*$, where $t_e^*, e = \{1, ..., n_e\}$ is the time instant at which evader e is captured.

Strategies for pursuers and evaders can be computed according to the objectives and characteristics of the game. In order to achieve capture, a pursuit policy is a function

$$g(y(t)) \triangleq p(t+1) = [x_1(t+1), ..., x_{n_p}(t+1)] \tag{6.2}$$

given the historical sequence of collected information until a time t, $y(t)$, and the desired position of all the $[1, \ldots, n_p]$ pursuers at a future time $t+1$:

$$p(t+1) \triangleq [x_1(t+1), ..., x_{n_p}(t+1)] \tag{6.3}$$

In the policy, the pursuers decide the next movement for the next discrete time instant.

Pursuit-evasion games can have the following characteristics:

- The games can be of type zero-sum, a competitive game where the interests of the players are in conflict and the strategies are optimal, or non-zero sum where cooperation takes place, some players win and others lose and the computed strategies are mixed (Vlassis, 2007).
- A game can contemplate a scenario with one pursuer-one evader, or multi-player with increased complexity of the strategies. Multi-agent systems theory concepts can be used in the solution of multi-player games (Liu and Jin, 2009).
- The games can be continuous, considering the models of the pursuer and evader for continuous motion, or discrete, considering discrete models of the possible navigation actions like in visibility-based solutions (Yavin, 1986), or in matrix representations (Shinar and Silberman, 1995; Gurel-Gurevich, 2009).
- Complete information (the history of the game is known) or imperfect information (the evader and the pursuer do not know the action selected by the opponent until

it has been implemented) can be available for the solution of the game. If the information is incomplete, predictions and models of the opponent can be used to try to solve the games (Gurel-Gurevich, 2009).

- A traditional pursuit-evasion game is zero-sum (competitive), with perfect information (fully observable, knowledge of past actions), deterministic and sequential. Optimal strategies can be computed for the pursuers and evaders. Some of the most common assumptions of traditional pursuit-evasion games are: all payoffs (cost functions) are known, all players are rational and the rules of the game are common knowledge (Isaacs, 1965; Gurel-Gurevich, 2009).
- A more modern approach of the games contemplate stochasticity of the elements and incomplete or imperfect information (due to sensing and its disadvantages in terms of noise and uncertainty in realistic applications, and the unavailability of prior models of the opponents), discrete and simultaneous (anticipating opponent, as opponent would also anticipate the selection of an action). Also, multi-player games have been popular. Cooperation and negotiation mechanisms lead to mixed strategies and non-zero sum results (Başar and Olsder, 1999; Vlassis, 2007; Gurel-Gurevich, 2009; Liu and Jin, 2009).
- In pursuit-evasion games, the evaders frequently act according to: (a) the maximisation of the time of capture by computing a control law, (b) reaction to the action of the pursuer from a controller or from sensor-based path planning, (c) a set of deterministic fixed rules, or (d) unpredictable intelligence (Lim et al., 2004).

Some of the principal variations of pursuit-evasion games involve different added problems, like limited information, more kinematic constraints, a need for short-term actions, cooperation and coordination:

- Homicidal chauffeur: considering steering capabilities of the pursuer and evader (Isaacs, 1965; Yavin, 1988; Miloh and Pachter, 1989; Glizer, 1999).
- Princess and monster (also called hunter and rabbit): considering different velocities for the pursuer and the evader (Isaacs, 1965; Adler et al., 2003).
- Lion and man: the environment is bounded, and the solution depends on the initial locations of the players (Karnad and Isler, 2008).
- Multi-pursuers: adding concepts of multi-agent systems (Liu and Jin, 2009).
- Herding: guiding the evader towards a specific location for its capture (Bopardikar et al., 2009; Khalafi and Toroghi, 2011).

Prey-pursuit games have been used for the design and control of autonomous robots and unmanned vehicles in different tasks (Vieira et al., 2009; Desouky and Schwartz, 2009; Vidal et al., 2002; Hespanha et al., 2000; Li and Cruz, 2007; Lachner et al., 1998; Vidal et al., 2001; Kim et al., 2001; Murrieta-Cid et al., 2007; Chung et al., 2011; Ni and Yang, 2011)), ship collision avoidance tasks (Miloh and Pachter, 1989), and the design of anti-missile defence strategies by intercepting the missiles (Shinar and Silberman, 1995; Yang and Yang, 1996;

Shinar and Shima, 1996; Lipman et al., 1997; Shinar and Shima, 2002; Shima and Shinar, 2002).

6.2 Solutions to pursuit-evasion games

Solutions to the games can be separated into two categories according to Isaacs (1965): (a) games of a kind, where the objective is finding if solutions to the pursuit-evasion game exist, i.e. if the evader can be captured by the pursuer, and under which circumstances; and (b) games of degree, where the objective is finding a strategy or solution for the pursuer or for the evader.

The main groups of solutions to pursuit-evasion games are presented in the following subsections (Luo et al., 2008):

- Control laws can be computed directly from differential games by using optimisation theory concepts, when the models of the environment, pursuers and evaders are known (Subsection 6.2.1). Controllers can also be computed from learning and interaction with the environment, using techniques from intelligent control theory when no models are available (Subsection 6.2.2).
- Co-evolved strategies for pursuit and evasion using genetic algorithms (Subsection 6.2.3).
- Visibility-based analysis starts by representing the problem as a graph (generally a grid), and then searching for solutions (Subsection 6.2.4).
- For more realistic applications to pursuit-evasion games, probabilistic analysis computes solutions considering the stochasticity of the information and the models of the environment, pursuers and evaders (Subsection 6.2.5).
- Solutions from a multi-agent systems perspective have been proposed when more than two players are contemplated in a pursuit-evasion game (Subsection 6.2.6).

6.2.1 Differential games

When models of the environment and the players are available (in the form of differential equations), a solution to the game can be computed by optimisation theory. The optimal strategies (or control laws) for the pursuer and the evader result from the corresponding minimisation or maximisation of a cost function related to the differential model (e.g., the capture time). The strategies can be computed for both players at the same time. For example, Leondes and Mons (1979) computed optimal strategies for pursuers with perfect information about the environment, and evaders with noisy measurements of the state and the environment. Yavin (1986) computed optimal strategies for the pursuer receiving noisy measurements of the location of the evader, and the evader with perfect information. Li and Cruz (2007) solved a stochastic differential pursuit-evasion game, with imperfect and partial information and extension to realistic applications in robotics. Bhattacharya et al.

(2009) computed strategies for the pursuer and the evader when obstacles are present, by incorporating the obstacles as part of the boundaries of the environment.

Strategies for one player can be computed if the strategy of the other player is known. For example, de Villiers et al. (1987) computed a feedback evasion law for the evader. Cherkasov and Yakushev (2002) computed an optimal navigation for an evader, considering a pursuer performing a proportional navigation strategy. Lim et al. (2004) and Shinar et al. (2010) performed a worst-case analysis to predict the behaviour of an intelligent evader, and then computed a control strategy for the pursuer. Abramyants et al. (2004) developed control laws for a pursuer that detects and classifies targets (as evaders could use false targets to distract a pursuer) and then captures the evader. Galloway et al. (2007) computed a control law for pursuit from stochastic differential equations, considering an evader with a specific steering control.

Other researchers have been interested in analysing the characteristics of the optimal solutions, like the conditions of capture (games of kind). Meier (1969) and Mizukami and Eguchi (1977) studied the capture region when the evader is faster than the pursuer using geometric methods. Yavin (1987) computed the optimal conditions for pursuit and the corresponding control laws when the evader uses a false target strategy to distract the pursuer. Bernhard and Pourtallier (1994) studied the feasibility of capture of an evader when getting information about it is costly in time for the pursuer. Cardaliaguet (1997) characterised the victory of each player in a pursuit-evasion game in a geometric form. Glizer (1999) considered the steering capabilities of the pursuer and evader and their speed to find if capture is possible for slower evaders.

Numerical solutions to the differential equations can be computed by means of a boundary value problem, and approximated with numerical methods (Pachter and Yavin, 1981; Yavin, 1988; Raivio and Ehtamo, 1997; Lachner et al., 1998; Hong-yan and Zhi-qiang, 2010) or neural networks (Pesch et al., 1995) trained from previously computed optimal strategies.

Current areas of research in differential games are (Buckdahn et al., 2011): the presence of state constraints; the use of stochastic differential equations; dealing with incomplete information; solutions to non-zero sum games; and using long-time average to reduce the complexity of the differential games. Biological inspiration has been used in recent years as part of the differential modelling. Fuchs et al. (2010) developed optimal control strategies for a pursuer and two evaders that cooperate (biologically inspired), by using a cost function that captures the cooperative behaviour of the evaders in a 2-D navigation plane. Hong-yan and Zhi-qiang (2010) modelled a pursuer and an evader as a fish school each (considering the minimum turning radius in the confrontation), solved with numerical methods.

6.2.2 Controllers

Other different control solutions (controllers) have been proposed. The use of intelligent controllers (fuzzy, neural networks) makes it possible to compute a pursuit strategy from the interaction of the pursuer with the evader and the environment through sensing (Lim et al., 2004). Jeong and Lee (1999) designed a fuzzy logic controller for pursuit from the optimisation provided by a genetic algorithm from the interaction with an evader. Desouky and Schwartz (2009) designed a fuzzy controller for pursuit that adjusts its parameters from interacting with the environment: an optimal strategy is computed first using Q-learning, and then it is used to compute the parameters of the controller with genetic algorithms, considering the Q-learning strategy as training data. Chung and Liu (2010) developed a steering fuzzy logic controller, with parameters computed from genetic algorithms, for a mobile robot working as a pursuer, for a trajectory of an evader known *a priori*.

6.2.3 Co-evolution of strategies

Pursuit and evasion strategies have been developed from co-evolution using genetic algorithms. The idea comes from a biological inspiration on predator-prey interactions (Miller and Cliff, 1994). Wahde and Nordahl (1998b) co-evolved pursuers and evaders represented as neural networks for environments with and without obstacles and boundaries, reporting the emergence of protean behaviours in the evaders. Kim and Tahk (2001) proposed formulating a pursuit-evasion game as a constrained min-max problem, and solving it by using Lagrange multipliers to handle the constraints and co-evolutionary computation. Tay et al. (2008) co-evolved pursuer and evader strategies (in the form of combinations of actions to form steering strategies) in an environment with boundaries and obstacles, using chemical genetic programming.

6.2.4 Visibility-based analysis (graph search)

The state space in which pursuers and evaders move can be represented as a graph, where lines and vertices indicate possible motions (Chung et al., 2011). Pursuers and evaders move along lines considering constraints on their velocities, visibility and obstacles in the environment. Some researchers have developed randomised strategies for pursuit. For example, Adler et al. (2003) and Dumitrescu et al. (2010) developed randomised algorithms in a grid (the shape of the graph) with hiding from the line of sight of the evaders, when both the pursuers and evaders have limited vision. Isler et al. (2005) developed randomised algorithms for the capture of an unpredictable evader in polygonal environments. Suzuki and Zylińsky (2008) designed randomised and deterministic algorithms in 3D applied to a robotic multi-pursuer game, including forming a trap for the evader, and hiding before capture.

On the other hand, visibility-based analysis has also been used to find the conditions of capture of the evader under different constraints like occupation of vertex and edges, lack of information about the evader, limited motion of the pursuers and different speeds (Dawes, 1992; Neufeld, 1996; Isler et al., 2006; Nisse and Suchan, 2008; Fomin et al., 2008; Fomin and Thilikos, 2008; Brass et al., 2009; Fomin et al., 2010; Scott and Stege, 2010). The results are often given in terms of the necessary number of pursuers for capture.

6.2.5 Probabilistic analysis

From a more realistic perspective, the representation of the players and the environment in the game can be built into a probabilistic model (probability map, Gaussian proccess, Markov decision process), considering incomplete information, noise and stochasticity coming from sensing or predictions (Chung et al., 2011). For probabilistic models of the evader and the environment, strategies of pursuit can be computed as a greedy search towards capture, or the strategies can be learned by reinforcement learning.

Ficici and Pollack (1999) computed a statistical model of the evader, then the pursuer computed a path towards the evader, whilst the evader evolved a neural network based controller. Hespanha et al. (2000) used receding horizon control to compute the optimal policies for a pursuer and an evader in a Markov modelled game. Ishiwaka et al. (2003) modelled a multi-pursuer game as a decentralised partially observable Markov decision process (POMDP), and each pursuer computes its own strategy with Q-learning. Chung and Furukawa (2009) proposed the use of particle filters to detect and model evaders in the environment, and then pursuers evaluate the information about the evaders to choose which evader to capture.

Other authors preferred computing a discrete probability map. Kim et al. (2001), Vidal et al. (2002) and Kim and Shim (2003) computed navigation greedy policies for robots from probability functions or maps. Antoniades et al. (2003) proposed computing probability maps of the evaders combined into a single reward function, dividing the map to avoid overlap of pursuing the same evaders, and then computing policies using search methods (within adjacent cells, or in the whole map to compute pursuit policies) in a multi-player game. Zheng et al. (2007) used a discrete (grid) probability map of the environment, and proposed a search algorithm to find and capture pursuers that avoids many visits to the same cells by pursuers. Huang et al. (2009) proposed building a probabilility map of the environment and the evader, computing a pursuit policy from searching the map considering the kinematic characteristics of the pursuer. Kwak and Kim (2010) used a probability map to represent a multi-player game in 2-D, using the maximisation of the probability of capture (global-max) as pursuit policies.

Table 6.1 shows a comparison of the main characteristics of the groups of solutions to pursuit-evasion games for single pursuer-single evader games, presented in the previous subsections.

Table 6.1: Comparison of groups of solutions to pursuit-evasion games for single agents.

FEATURE	DIFFERENTIAL GAMES	VISIBILITY-BASED ANALYSIS	PROBABILISTIC ANALYSIS	CO-EVOLUTION	CONTROLLERS
Model	Differential equations that model dynamics of players	Graph of configurations or topology	Probabilistic (e.g. probability map, Gaussian process)	Basic model (e.g. neural network	Optional
How to solve the game	Optimisation theory: minimisation or maximisation of a cost function subject to constraints	Optimal search	Greedy search (maximising probabilities), or reinforcement learning	Evolutionary and genetic algorithms	Learning from interaction, training
Solution	Control laws for pursuer and evader, or one of both if the other is known	Random strategy for pursuit according to constraints	Greedy strategy or policy for pursuit according to available information (e.g. sensed)	Controllers for pursuit and evasion	Controller for pursuit or evasion
Other solutions	Capture and evasion regions and limits, numerical solution of the control laws	Number of pursuers for capture an evader with particular characteristics, visibility regions			

6.2.6 Multi-agent systems approach

The extension of one pursuer-one evader games to multi-players was introduced in the 1980s by Benda (Liu and Jin, 2009). Many of these games consider finding pursuit strategies from a multi-pursuer perspective, rather than evasion. Different combinations of elements, from single agent groups of solutions and multi-agent systems, have been used to solve multi-player games:

- Incorporating concepts of multi-agent systems to the solution of the game, particularly communication between players, cooperation and negotiation, coalitions. Examples of the use of communications are: communication of information from sensing the environment (Vidal et al., 2001; Zhao and Jin, 2005; Fan et al., 2009), mixing direct and indirect (sharing rewards) communication between pursuers (Sun et al., 2003) and sharing individually computed policies (Zhao and Jin, 2005). Other examples of coordination through negotiation techniques and forming coalitions are: voting to combine individual decisions of the pursuers into a joint pursuit strategy (Partalas et al., 2007), a bidding mechanism to form coalitions of pursuers to chase an evader collectively (Cai et al., 2008) and the formation of alliances between the pursuers to chase a detected evader in formation, controlling their navigation strategies through a neural network (Ni and Yang, 2011).

- Using multi-agent reinforcement learning to compute optimal strategies for pursuit from stochastic models of the environment. Tamakoshi and Ishii (2001) used Q-learning with function approximation in the form of a normalised Gaussian network for the strategies of the pursuers. Quian and Hirata (2003) proposed a Q-learning automaton algorithm to compute the strategies for the pursuers. Kaya and Alhajj (2002), Gültekin and Arslan (2002), Kaya and Alhajj (2003), Kaya and Alhajj (2004) and Watanabe and Takahashi (2007) estimated a fuzzy model of other pursuers and evaders, and computed the best strategy using Q-learning. Ishikawa et al. (2008) proposed a model of internal rewards to guide the strategies of the pursuers.

  Also, multi-player pursuit-evasion games has been one of the popular testbeds for multi-agent reinforcement learning algorithms (Panait and Luke, 2005). Kilic and Arslan (2002) and Duman et al. (2005) proposed a fuzzy version of Q-learning for and evaluated it for a pursuit task. Meng et al. (2005) proposed a combination of Nash-Q-learning, correlated equilibrium (CE) Q-learning and hill-climbing algorithms and evaluated it through pursuit strategies.

- Decomposing the multi-player game into simpler ones of one pursuer-one evader or various pursuers-one evader. For example, Li and Cruz (2006) and Li et al. (2007) computed suboptimal solutions from differential equations iteratively over limited look-ahead intervals. Wang et al. (2002) divided a multi-player game into multi-pursuer single-evader games, using a formation control law to surround an evader and avoid collisions between pursuers. Schenato et al. (2005) proposed the use of a sensor network to predict the behaviour of the evaders and assign an evader to a pursuer, then each pursuer plans a path to avoid collisions and capture the corresponding evader. Cai et al. (2009) proposed the formation of different coalitions of pursuers that would chase a single evader each. Vieira et al. (2009) proposed decomposing a multi-player game into various multi-pursuer single-evader games, solved by a graph search approach and considering a communication network for coordination within the multi-pursuers. Kwak and Kim (2010) divided a probability map so that a pursuer would focus on an evader accordingly.

- Other approaches for multi-pursuers include the combination of intelligent control concepts, swarms and formations and geometrical analysis. Nitschke (2003) evolved neural network controllers to compute cooperative behaviours for pursuit. Kopparty and Ravishankar (2005) computed lines towards the evader and avoiding obstacles in a multi-pursuer setting (geometrical algorithms). Bopardikar et al. (2007) used multi-pursuers maintaining a formation (according to their detection radius) to perform a sweeping pattern to find an evader, and then change the formation to capture it. Hládek et al. (2008) proposed a centralised entity giving suggestions for the locations of the pursuers according to a fuzzy model of the environment. Jin and Qu (2010) coordinated the locations of the pursuers as a polygon formation (intersecting circles of capture) to surround a faster evader. Bopardikar et al. (2009) also used formations of pursuers with limited sensing capabilities to herd evaders. Fan et al. (2009) used communication of last observed locations of evaders and ant inspired motions (ant swarms) to follow evaders with mini-robots (e-Puck) and considering their limited sensing capabilities.

The next section reviews the topic of reinforcement learning, previously mentioned as part of the computation of strategies and control systems for pursuit in a pursuit-evasion context.

6.3 Reinforcement learning

The concept of reinforcement learning comes from psychology studies on the process of conditioning, where a behaviour is favoured if a stimulus is given after its performance. This idea has been adapted to machine learning, where a decision maker (also frequently called *agent*) learns to select actions that maximise a reward given by the environment (Sutton and Barto, 1998). Reinforcement learning has been applied to solve several kinds of problems in artificial intelligence, game theory, control, decision making and planning, among others.

The milestones of reinforcement learning are presented in Table 6.2, with emphasis on the different mathematical models. The mathematical background of reinforcement learning comes from its modelling as a Markov decision process (MDP), numerically solved using dynamic programming algorithms.

A discrete MDP is defined as a triplet $< \mathcal{S}, \mathcal{A}, P_0 >$, where $\mathcal{S}$ is the set of possible discrete states and $\mathcal{A}$ is the discrete set of actions performed by the decision maker. $\mathcal{P}_0$ is the state transition probability kernel, which assigns to each state-action pair, $(s, a) \in \mathcal{S} \times \mathcal{A}$,

Table 6.2: Milestones of reinforcement learning

YEAR	CONTRIBUTIONS
1911	Studies on trial and error learning by Thorndike (Sutton and Barto, 1998).
1950-1959	Bellman equation is developed in control theory as a relation between the dynamics of the system and a value function. Markov decision processes (MDPs) are introduced. Trial and error learning is studied in computer science by Minsky. Learning with temporal-difference is developed by Samuel (Sutton and Barto, 1998).
1960-1969	The policy iteration method is proposed to solve MDPs. Reinforcement learning is formally used in engineering (as part of artificial intelligence) for credit assignment problems (Sutton and Barto, 1998).
1970-1979	Learning automata are developed. Reinforcement learning and temporal-difference are joined by Klopf and Sutton. Classifier systems are developed by Holland. The actor-critic architecture is proposed. Learning for neural networks is done with reinforcement learning. Temporal-difference is used as a prediction method separated from control systems. Q-learning is proposed by Watkins (Sutton and Barto, 1998).

a probability measure over $\mathcal{S} \times \mathbb{R}$. The state transition probability kernel assigns: *(a)* a probability for a transition from state s to a new state, s', when choosing an action a, and *(b)* an immediate reward function, $\rho : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which assigns a reward when choosing a determinate action in a determinate state in a discrete instant of time (Sutton and Barto, 1998; Hespanha et al., 2000; Buşoniu et al., 2006; Vlassis, 2007; Szepesvári, 2010).

When performing action a, the state of the environment changes from s to s' in a time step t with a probability of $P(s', s, a)$, and receives a reward

$$r(t) = \rho(s', s, a) \tag{6.4}$$

The actions are selected according to a policy, π, that can be deterministic or stochastic. A deterministic policy is a direct mapping from states to actions $\pi : \mathcal{S} \to \mathcal{A}$, and a stochastic policy is a mapping to probability distributions $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$.

Thus, the problem of learning is finding a policy, $\pi : s \to a$, such that starting at state s_0, and selecting a series of actions, a goal is reached in a minimum of i steps. A discounted reward is maximised over time,

$$R(t) = \sum_{i=0}^{\infty} \gamma^i r(t + i + 1) \tag{6.5}$$

where $\gamma \in (0, 1)$ is the discount rate. To find the optimal policy in terms of the reward, iterative methods from dynamic programming using Bellman optimality equations are used. A value function for $\mathcal{S}$ (the states) can be defined as an accumulation of rewards obtained when following a policy π given a starting state, s_0,

$$V^{\pi}(s) = E_{\pi}\left\{R(t)|s\right\} = E_{\pi}\left\{\sum_{j=0}^{\infty} \gamma^j r(t + j + 1)|s\right\} \tag{6.6}$$

where E_{π} is the expected value. Likewise, a value function for the taken action in s when

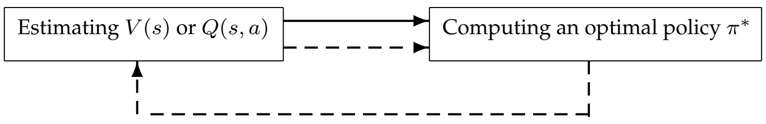

Estimating $V(s)$ or $Q(s,a)$
Computing an optimal policy π^*

Figure 6.1: The reinforcement learning problem: estimating the state or action-state values, and then computing an optimal policy. Solid line: estimation of the value functions first, and then computing the optimal policy. Dashed line: estimation and computing the policy iteratively in time.

following a policy π can be defined as

$$Q^\pi(s,a) = E_\pi\left\{R(t)|s,a\right\} = E_\pi\left\{\sum_{j=0}^{\infty}\gamma^j r(t+j+1)|s,a\right\} \tag{6.7}$$

The Bellman equations (or dynamic programming equations) give a value to the states or action-state pairs from the expected reward, recursively (in terms of the future actions),

$$V^\pi(s) = \sum_{a\in\mathcal{A}(s)}\pi(s,a)\sum_{s'\in\mathcal{S}'}P(s',s,a)\left[\rho(s',s,a)+\gamma V^\pi(s')\right] \tag{6.8}$$

$$Q^\pi(s,a) = \sum_{s'\in\mathcal{S}'}P(s',s,a)\left[\rho(s',s,a)+\gamma V^\pi(s')\right] \tag{6.9}$$

The optimal policies that maximise the value or action-value functions can be defined as

$$V^*(s) = \max_\pi V^\pi(s) \tag{6.10}$$

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) \tag{6.11}$$

Substituting the optimal values in the Bellman equations, results on the Bellman optimality equations,

$$V^*(s) = \max_{a\in\mathcal{A}(s)}\sum_{s'}P(s',s,a)\left[\rho(s',s,a)+\gamma V^*(s')\right] \tag{6.12}$$

$$Q^*(s,a) = \max_{a\in\mathcal{A}(s)}\sum_{s'}P(s',s,a)\left[\rho(s',s,a)+\gamma\max_{a'\in\mathcal{A}(s')}Q^*(s',a')\right] \tag{6.13}$$

Thus, the optimal policy is a set of actions that delivers the optimal value or action-value function, from the rewards obtained from the transitions to future states. When the value or action-value functions are not known, the problem becomes finding an estimation or prediction of the functions, and then computing the optimal policy, as shown in Figure 6.1. The policies can be deterministic (selecting an action for a state) or stochastic (as probabilities of selecting actions for a state).

Reinforcement learning is different from other forms of learning for the following reasons

(Sutton and Barto, 1998; Smart and Kaelbling, 2002):

- Reinforcement learning is used when no analytic solution of the model of the environment is available, if the environment is a model of the reality, or when environment information is available only through interaction with it.
- No knowledge of the MDP is needed to solve the problem of reinforcement learning, only the rewards.
- Exploration and exploitation take place in the learning, according to the requirements of design.
- Reinforcement learning can be applied to solve stochastic problems.
- Algorithms for online (learning while performing real actions with the current policy) and offline learning (learning first, then performing real actions accordingly) are available.
- Dealing with different amounts of data is possible through generalisation, sampling and approximation (using neural networks to calculate and store the value functions, instead of a look-up table for smaller state or action-state space).
- Convergence to an optimal solution has been proved for some algorithms, like Q-learning for single agent cases and MDP (Sutton and Barto, 1998; Gosavi, 2008).
- In robotics, reinforcement learning allows computing a high level policy with the right formulation of the reward.

6.4 Reinforcement learning algorithms

Reinforcement learning algorithms consider the mathematical theory presented in the previous section, evaluating the selection of actions for specific states to compute an optimal policy. Dynamic programming allows performing value iteration and policy iteration to compute the evaluation of a policy and the computation of a new one, respectively, for several episodes and steps recursively. An episode consists of a series of steps (state $\rightarrow$ action $\rightarrow$ reward) from a initial condition to a terminal condition or state (Szepesvári, 2010).

The different proposed algorithms try to compute good approximations of the value functions, and optimal policies according to the requirements of the task, in terms of availability of models, available information from the environment, computational cost, required speed of the deliberation, risks of exploration of actions, among others. The main differences between the algorithms are: how to compute the value functions (using future predictions, considering visited states or action-state pairs), when to update the current policy, dealing with the curse of dimensionality and large sets of data and if the algorithm is for one agent or many (multi-agent reinforcement learning).

6.4.1 Monte Carlo

In Monte Carlo algorithms, the value or action-value functions are estimated as an average of the results after each episode in the game. The every-visit Monte Carlo algorithm computes the value function evaluation for an episode (all the steps, visiting states and actions in multiple occasions), while first-visit Monte Carlo computes the value function evaluation after the first visit of all the states and actions. Monte Carlo algorithms do not perform any prediction of future actions and state transitions (Sutton and Barto, 1998; Szepesvári, 2010).

6.4.2 Temporal difference (TD)

Temporal difference algorithms use predictions (bootstrap) to compute the estimate of the value or action-value functions ($V(s)$ or $Q(s,a)$). The algorithms can be offline (or off-policy) when the value or action-value function of a policy is being estimated while the policy changes, or online (or on-policy) when a policy that is being evaluated (through computing and estimate of its value or action-value function) (Sutton and Barto, 1998; Gosavi, 2008; Szepesvári, 2010).

TD(0) algorithms, like Q-learning and Sarsa (shown in Algorithms 3 and 4) consider only the next step for their bootstrapping (single step predictions), while TD(λ) algorithms consider several steps in the bootstrap (multiple step predictions). $\lambda \in [0,1]$ is a weight for the contribution of the rewards resulting from every transition of states at each step (Sutton and Barto, 1998; Peeters et al., 2007). In Monte Carlo, $\lambda = 1$, and in TD(0), $\lambda = 0$.

The main algorithms within temporal difference are (Sutton and Barto, 1998; Gosavi, 2008; Szepesvári, 2010):

1. **Q-learning:** an algorithm that does not require a stochastic model of the environment, but only knowledge about the rewards and the observation of the transitions to the next state. It is useful for non-stationary systems and goals. The policy is updated offline, as the used policy could change while other is being evaluated (the value function is being computed). Q-learning has been widely used throughout the literature due to the simplicity of the value function updates, but its major drawback is the extrapolation of the algorithm when dealing with a continuous action-state space by means of function approximation. The tabular version for the Q-learning algorithm is shown below, using α as the learning rate (determining the speed of the processing), γ the discount rate (weighting the previously received rewards) and $Q(s',a')$ as the action-state value for the actions for the next predicted state. The tabular version of Q-learning is presented in Algorithm 3.

2. **Sarsa:** an online method, where the evaluated policy (by computing its value function) is unique during the length of the evaluation. This algorithm is a contemporary of Q-learning. The tabular version of the Sarsa algorithm is presented in Algorithm 4, for the same learning parameters as Q-learning.

3. **Actor-critic architecture:** the critic performs an evaluation of a policy (computing

Algorithm 3 Q-learning

```
Initialise $Q(s,a)$ with zeros
repeat
  Initial state $s$
  repeat
    Select action $a$ for $s$ using an action selection method
    Simulate $a$, obtain reward $r$ and observe next state $s'$
    Update: $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s',a')]$
    $s \leftarrow s'$
  until $s$ is the goal
until a number of $iterations$
The policy is selected from $\max_{a \in \mathcal{A}(s)} Q(s,a)$ for all $s \in \mathcal{S}$
```

Algorithm 4 Sarsa

```
Initialise $Q(s,a)$ with arbitrary values
repeat
  Initial state $s$
  repeat
    Select action $a$ for $s$ using an action selection method
    Simulate $a$, obtain reward $r$ and observe next state $s'$
    Select action $a'$ for $s'$ using an action selection method
    Update: $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma Q(s',a')]$
    $s \leftarrow s'$
  until $s$ is the goal
until A number of $iterations$
```

the value or action-value function) using algorithms like Sarsa(λ) or least-square temporal difference-Q(λ). The actor improves the policy in a greedy manner from the computed function by the critic (where the best actions for a state are the maximisation of $V(s)$ or $Q(s,a)$), or including some exploration using ϵ-greedy or Boltzmann probabilities to select actions randomly. This architecture is useful in partial observable domains, to compute stochastic policies.

6.4.3 Eligibility traces and function approximation

Eligibility traces allow modulating the effect of the history or the predictions of visited states and action-state pairs, by assigning weights (Sutton and Barto, 1998; Szepesvári, 2010). Eligibility traces can be accumulative, or replaced by a fixed value after a number of occurences. They speed up the learning, and they can be applied to both tabular and function approximation based algorithms.

When the state space is large or infinite, the value or action-value functions can be approximated by a set of parameters (weights) and features (basis functions) that map the action-state pairs, for example

$$Q(s,a) = \omega^T \psi \tag{6.14}$$

where ω^T is the transpose of the weights, and ψ is the set of basis functions. The basis functions that can be employed include polynomial, Fourier, wavelet, tensor product constructions (like radial basis functions), binary features (like state aggregation) and tile coding. Gradient temporal difference algorithms can be used to compute the estimate of the value or state-value functions to guarantee the convergence to the optimal values.

When the dimensionality of the state space, $\mathcal{S}$, is large (*curse of dimensionality*), interpolators and function approximations can be used including: discretisation of the continuous state spaces (Santamaria et al., 1997), neural networks (ten Hagen and Kröse, 2000; Gosavi, 2008), fuzzy logic (Berenji, 1994; Glorennec, 1994), kernel smoothing methods (Gosavi, 2008) and decision trees (Szepesvári, 2010).

Other approaches to solve the problems of dimensionality, amount of data, different timing of processes, and decentralised processing include hierarchical reinforcement learning and multi-agent reinforcement learning algorithms. Hierarchical architectures have been proposed to deal with temporal abstractions, where the time between a decision and the next is a random variable (semi-Markov decision processes, SMDP). Q-learning and Sarsa algorithms can be applied to solve SMDP, along with the use of hierarchies consisting of macro-actions for large scale planning and problem solving. The use of macro-actions allows the computation of partial policies that rule more primitive actions (subsets of the set of states, $\mathcal{S}$), and that are valid for different durations of time. Hierarchical reinforcement learning can be applied to simultaneous processes (concurrent actions), coordination of multi-agent systems, and to preserve the memory (in a structural form) of sequences of actions (Barto and Mahadevan, 2003).

Multi-agent reinforcement learning algorithms consider decentralised and hybrid computation of policies, complemented by the use of coordination techniques (negotiation, cooperation, alliances) and indirect and direct communication (e.g., distributed rewards, sharing knowledge). Some examples of multi-agent reinforcement learning applications to multi-robot navigation have been mentioned in Section 6.3. Current challenges to target in multi-agent reinforcement learning algorithms include dealing with partial observable Markov decision processes (POMDP) and the lack of information about the environment, the presence of irrational opponents, and continuous and large sets of information (Yang and Gu, 2004).

6.4.4 Variations in the key features of the algorithms

Real life applications call for dealing with the curse of dimensionality and continuous supply of information, and also to find the best combination of parameters in an algorithm to provide a solution to a specific task. The following paragraphs present a summary of the possible variations proposed in the literature for the reinforcement learning algorithms. The parameters that need to be selected include the action selection mechanism, the balance between exploration and exploitation in the search for the optimal policy, other learning parameters (including the learning rate and discount rate), and the reward function.

Action selection mechanisms

This stage in a reinforcement learning algorithm allows a degree of exploration of the possible actions in the steps of an episode, to find an optimal policy from interactive learning. Exploration of alternatives allows maximising the knowledge and identifying a system. A balance between exploration and exploitation is important, as it allows computing other policies that could lead to the global optimal as a result of the exploration, and at the same time produces policies that are useful for a task. Three main action selection mechanisms have been recurrently used (Sutton and Barto, 1998; Gosavi, 2008; Szepesvári, 2010):

- **Greedy action selection:** this method selects the action with the highest value for the current or future state,
$$a^* = \max_{a \in \mathcal{A}(s)} Q(s,a) \tag{6.15}$$
where a^* is the optimal action selected from the best action values of all possible actions for that state s. The method performs exploitation and does not leave room for exploration, which could lead to local maxima problems and the computation of suboptimal strategies. For this reason, although it is more computationally costly, exploration is an essential part of the other action selection methods.

- **ϵ-greedy action selection:** a probability ϵ is used to balance the exploration and exploitation, being decreased as the process of learning evolves towards the optimal policy. The corresponding action for state s is defined as
$$a = \begin{cases} a^* & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases} \tag{6.16}$$
where a^* is the optimal action from Equation 6.15 and ϵ is limited to the range $[0,1]$.

- **Soft-max action selection:** the ϵ-greedy method chooses equally between the actions for the current state, which could be inconvenient as the worst actions (in terms of their values) can be selected with the same probability as the best ones. To overcome this situation, the soft-max action selection method distributes probabilities to all possible actions for a state, ranked according to their utility (Q value), based on a Gibbs or Boltzmann distribution:
$$P(a) = \frac{e^{Q(s,a)/\tau}}{\sum_{b \in \mathcal{A}(s)} e^{Q(s,b)/\tau}} \tag{6.17}$$
where $P(a)$ is the probability for an action a from the set of possible actions for state s, $b \in \mathcal{A}(s)$; and τ is a varying parameter known as the temperature, that modifies the randomness of the action selection. High temperatures allow exploration, and values closer to zero lead to exploitation.

Learning parameters

Learning parameters also help to deal with balancing the exploration and exploitation, and provide the settings for the learning (the speed of the learning, and sometimes the convergence to a solution). Static parameters have been traditionally used throughout the literature, but dynamic parameters allow more control of the learning process.

- **Fixed learning parameters:** traditionally, the learning parameters (like the learning and discount rates) are normally set to a fixed value within a range, and they are not modified during the process of learning. Some optimal ranges for the parameters have been suggested in popular literature (Sutton and Barto, 1998). Normal ranges of parameters are: $0 \leq \gamma \leq 1$ for the discount rate, and $0 \leq \alpha \leq 1$ for the learning rate. Small values of the learning rate, α, and the discount factor, γ, allow a great degree of exploration, whilst values of almost 1 imply exploitation. For the action selection mechanism, a large value of ϵ (within $[0, 1]$) or a large temperature, τ, cause a random behaviour in selecting actions.
- **Dynamic learning parameters:** the principal target of using dynamic learning parameters has been favouring exploration at the beginning and then transitioning smoothly towards merely exploitation by reducing the rates. Proposed forms have been used to modify the learning parameters along the with learning, for example a selection by fuzzy logic according to the needs of exploration or exploitation (Akbarzadeh et al., 2003), exponential decay functions to transition the parameters (particularly the learning rate and ϵ or τ in the action selection mechanism) from exploration towards exploitation (Sutton and Barto, 1998), and genetic algorithms to compute the best values for all the tabular Q-learning parameters in a specific task (Kamei and Ishikawa, 2004, 2006).

Reward functions

The design of the reward function for the learning is an important part of a reinforcement learning algorithm, as it can speed up the process of computing the estimate of the value or action-value functions and thus improving a policy (or strategy). Shaping the reward function can be done by incorporating prior knowledge about the task, incorporating other estimators and sources of information (internal or external), and even decomposing the reward function as needed (Matarić, 1994; Laud and DeJong, 2003; Marthi, 2007).

6.4.5 Applications of reinforcement learning to navigation

Reinforcement learning has been used to compute navigation strategies (as another form to perform path planning and control) for robotics based on its previously mentioned characteristics. For example, ten Hagen and Kröse (2000) used an actor-critic configuration for a robot following a line without information about the system, based on a one layer feed forward artificial neural network function approximation. Smart and Kaelbling (2002) proposed computing the approximated value function (neural network) of the action-state space first with a fixed policy, and then upgrading the policy for autonomous navigation from learning via Q-learning. Yang et al. (2004) proposed the use of Q-learning with neural network based function approximation for the navigation of a mobile robot, where the weights and parameters of the neural network are computed from the interaction with the environment. Altahhan et al. (2008) used Sarsa (λ) with function approximation based on radial basis functions to compute a navigation strategy from visual information of the environment. Jaradat et al. (2011) proposed using Q-learning to train a robot for navigation in dynamic environments by limiting the number of possible states.

Also, an extension of multi-agent reinforcement learning algorithms is their application in multi-robot navigation tasks, like RoboCup, area sweeping, tracking targets, multi-pursuit (mentioned in Section 6.1) and transportation (Buşoniu et al., 2008). Some examples that compute navigation strategies (or policies) include: Fujii et al. (1998) employed the communication of sensed information about the environment from a robot to the other, then learning navigation actions to avoid collisions between robots; Asada et al. (1999) used system identification techniques (local predictive models) embedded in a robot to learn the actions of other robots and then to compute a navigation strategy with Q-learning; Melo and Ribeiro (2008) proposed an estimation of the policies of other robots from history, and the formation of a joint control navigation policy (from individual decisions) sharing the same reward function.

6.5 Discussion

A tendency to compute strategies for the pursuit and not the evasion was observed in the literature review of the present chapter. This situation is evident throughout the different solutions to the games, but is particularly evident in the multi-player case. For example, the use of cooperation to achieve pursuit, or the use of reinforcement learning, where distributed and joint strategies are computed mostly for the pursuit, or pursuit is used to evaluate the multi-agent reinforcement learning algorithms. Different examples of the design of strategies for the pursuers were mentioned throughout Section 6.2, as well as few exceptions towards the design of evaders and other attempts (e.g., Antoniades et al. (2003), Cherkasov and Yakushev (2002)). Other efforts on the design of evasion are presented in Section 3.2 using path planning techniques, in Subsection 6.2.1 as part of the solution to differential equations for both parties and in Subsection 6.2.3 as part of the co-evolution of pursuers and evaders. The result of differential games, visibility-based analysis, co-evolution and other multi-agent systems approaches lead to pursuit controllers or strategies developed for fixed evasion strategies, or the other way around, leaving little room for adaptation to radical changes in the opponent (maybe the result of intelligence, learning, evolution and adaptation). The evaders commonly do not have an adaptive control system that allows navigation with collision avoidance, and simultaneously avoiding capture by multiple pursuers. Also, the multi-evader case with fewer pursuers has not been studied as extensively as the multi-pursuit case, representing future opportunities to extend the research of this thesis.

Games of kind, in combination with visibility-based analysis, provide a tool to evaluate the feasibility of the evasion according to the capabilities of the pursuers. Incorporating the bio-inspired ideas of refuges and proteanism into the navigation strategies of the evaders could be reflected in the analysis of the necessary conditions to avoid capture. Nevertheless, the strategies used by the pursuers are not known, or could not be possible to predict accurately.

In robotic systems, the information provided can be incomplete and dynamic, as it is acquired from sensing, and the environment can change. Then, a stochastic model of the environment becomes a better option than a precise deterministic model (Chung et al., 2011). Sometimes, when designing strategies for the navigation of robots and other real life applications, only one side of the strategies or behaviours (pursuers or evaders) is developed and the other part is expected to be unknown along with the rest of the environment. In this thesis, the selection of reinforcement learning to solve a pursuit-evasion game from the evasion perspective contemplates both situations: the availability of stochastic information of the system, from locally sensed sparse information, and the interest of developing only the strategies for the evader from a worst-case scenario with respect to the pursuers as their real capabilities and strategies are unknown. The proposed solution to a pursuit-evasion game using reinforcement learning does not end with computed strategies for both the pursuer and the evader, but only for the latter one.

Reinforcement learning sometimes has been used to learn navigation strategies for robotics, instead of conventional path planning techniques, due to its advantages. Reinforcement learning does not require a complex model of the environment, but information about the allowed actions, the transitions of the states and the rewards provided by the environment after performing an action. Considering a navigation scenario in 2D, as in Chapter 4, the actions are adjacent cells in the free space, the state transitions according to the motion of the robot and the predictions from the pursuers, and the rewards can be extended from the proposed analysis of the environment in Section 4.4. A tabular reinforcement learning approach can be used considering the characteristics of the proposed analysis of the environment, but it could be updated to a function approximation if the action-state space increases (with a finer sensing, and for more precise allowed motions). The expected result of the combination of the concepts of the pursuit-evasion game, with reinforcement learning, is a learned navigation strategy for the evader, that will be updated according to the changes in the environment, and the available processing capabilities.

Deliberating a navigation policy for the problem in this thesis is performed as a simulated process of interaction with the environment, where the learning happens from the state transitions predicted considering a worst-case scenario for the pursuers and the motion of the robot. Even in real life implementations, the learning (computation of a navigation policy) cannot be performed from trial and error from direct interaction, as robots could not afford to be captured and the tasks cannot be restarted as simulations do. Also, the parameters of the learning that affect the speed of deliberation (learning rate, discount rate, reward functions, episodes, terminal state) need to be selected in such a way that the implementation can be used realistically, possibly at an optimality cost. All these considerations are used for the designs presented in the next chapter, where the same problem of navigation and escape from pursuers as in Chapter 4 is solved through navigation plans drawn from a tabular Q-learning algorithm. Q-learning has been selected due to its popularity for robotic navigation tasks.

6.6 Concluding remarks

The chapter presented a brief literature review on pursuit-evasion games, mentioning a general definition, their main characteristics, and the principal methods used to solve the games considering variations in the characteristics of the pursuers, evaders and environment. Many examples were given to illustrate the different methods, highlighting the lack of design of evasion strategies independently from the design of pursuit strategies.

Subsequently, the chapter continued with a brief literature review of reinforcement learning, introducing the mathematical background for the computation of optimal policies from the value functions of selecting actions and receiving rewards or punishments from the environment due to the consequences of that action. The most popular algorithms were mentioned, their extensions from discrete to continuous spaces for more realistic applications, and key aspects to select (parameters) within the algorithms. Examples of reinforcement learning applied to robot navigation and in a multi-robot (multi-agent) context were presented.

The literature review of this chapter focused in general on the following aspects:

- Analysing the characteristics of intelligent evaders involved in pursuit-evasion games, and how their strategies for navigation are designed independently from pursuers, to find inspiration for the designs presented in this thesis.
- Analysing examples of realistic applications of pursuit-evasion games to autonomous robots, including the limitations on the sensing and available information.
- Finding about the use of reinforcement learning for the computation of autonomous navigation strategies for robots, specifically for evasion (escape from pursuers) tasks.
- How to modify a reinforcement learning algorithm to make it more efficient in real life tasks, particularly for the problem of navigation and escape to solve in this thesis.

Previous and current research has focused on pursuit, or pursuit and evasion depending on the knowledge or predictions of each other. Evasion individually from pursuit, and without available information from the opponent (the pursuer) has not been studied with the same interest.

Finally, a discussion of the literature review was performed, with emphasis on the findings about the design of evaders independently from pursuers, the advantages and disadvantages of the available methods to solve pursuit-evasion games in the context of autonomous navigation for robots, and the advantages of the use of reinforcement learning for computing navigation strategies (policies) that include the escape from pursuers and other challenges of that implementation.

Reinforcement learning allows the computation of strategies (or control laws) from exploration of actions (noise). Some reinforcement learning algorithms do not need a model of the system to work, as interaction and reward functions are enough to determine the best strategy to perform. This aspect is important to consider in the design of control systems when the strategy of the pursuers is completely unknown and a model is nor available, as learning to avoid collisions and escape simultaneously could be achieved. Many variations of reinforcement learning algorithms with particular characteristics are available to choose from according to the requirements of the task.

The next chapter presents the formulation of a pursuit-evasion game from the evasion perspective, and considering the problem of navigation and escape to be solved throughout this thesis. Subsequently, the computation of a navigation policy (or strategy) based on tabular Q-learning with dynamic learning parameters is proposed, derived from the analysis of the environment presented in Chapter 4 and as a substitute for the path planning approach presented previously.

Chapter 7

Autonomous navigation and escape using reinforcement learning

Another way to perform the computation of solutions to the pursuit evasion game described in Section 4.1 is presented in this chapter, from the concepts presented in Chapter 6. Learning navigation strategies for a problem, where a model of the environment is not available, is possible through reinforcement learning.

Reinforcement learning has been previously used to solve some autonomous navigation problems (in the sense of collision avoidance) for previously unknown environments, due to its comparative advantages. These include dealing with environments whose mathematical model is unknown, but observations about the transitions as a consequence of actions and obtained rewards can be used to obtain information to learn an optimal strategy.

Also, the use of reinforcement learning is of interest as it has been previously applied to solve pursuit-evasion games, but mostly from the pursuit perspective. Exploring its counterpart, evasion, is proposed as another form to offer solutions to the problem of navigation and escape from pursuers. Section 7.1 complements the definition of the problem given in Section 4.1, from a game theory and reinforcement learning perspective.

Section 7.2 presents the design of solutions to the problem from learning navigation strategies. The section starts with the presentation of the analysis of the environment in Subsection 7.2.1, a complement from the one proposed for path planning techniques in Chapter 4. Subsequently, all the chosen parameters and features in combination with a tabular Q-learning algorithm, for the reinforcement learning approach, are presented in Subsection 7.2.2. The chosen features include: the action selection mechanism for the learning process; the learning parameters such as the learning rate, discount factor, exploration-exploitation balance, and number of iterations; the shaping of the reward function from a biologically inspired perspective; and the policy improvement process. The proposed ranges of operation of the parameters are evaluated experimentally in Subsection 7.2.3, for a single static environment, to reassure the computation of convergent strategies (also called policies) in a reasonable amount of time.

Section 7.3 evaluates the proposed Q-learning algorithm through simulated Monte Carlo simulations with randomly generated environments. The main feature to evaluate is the computation of navigation strategies that try to prevent the capture, when the sensing of the environment is updated. The statistical success of the resulting strategies when updating the sensing of the environment and simulating the motion of pursuers is evaluated. The simulations replicate the statistical analysis performed in Chapter 5 for the path planning approach. Visual examples of the computed strategies are presented to illustrate the computations.

Finally, Section 7.4 presents a discussion highlighting the main characteristics of the proposed solutions to the problem of navigation and escape from pursuit using reinforcement learning, and the main results from the simulations. Section 7.5 presents the concluding remarks of the chapter.

7.1 The problem from a game theory and reinforcement learning perspective

Considering the definition of the problem of navigation and escape from pursuers in Section 4.1, the contained elements in the environment can be extended to a game form:

1. **Environment**: the locally observed terrain with n_o obstacles and n_p pursuers, E. The environment is stochastic regarding the strategies adopted by the pursuers, as only some assumptions can be made about their behaviour, but without the certainty that they are true. The environment can extend beyond the currently observed part.
2. **Obstacles:** assumed to be static, the robot needs to avoid collisions with the obstacles.
3. **Evader:** the robot that wants to escape, by finding a navigation strategy according to the other elements in the environment.
4. **Pursuers:** assumed to be stochastic. No previous accurate model that describes their real behaviour is known, but selecting a specific model from the literature could be used to predict them in the computation of the strategies (e.g., worst-case scenario).

The information of the environment is assumed to come from a sensing and identification process. The approximate relative location of the obstacles and pursuers in the local environment is known (with or without absolute certainty, represented by a probability) if they are within the range of sensing and not obstructed by others. The relative location is given in terms of the distances from the elements to the robot.

The main characteristics of a game, according to the problem, are:

- The objective of the game is computing an evasion strategy according to the available information.
- The game is characterised by multiple pursuers, a single evader and multiple static obstacles.
- The information is incomplete (partially observable), as only some elements are observed during the sensing, and the information might not be as accurate as in games with complete information.
- Pursuers and the evader compute and execute their strategies simultaneously. A worst-case analysis can be considered for the prediction of pursuers and state transitions.
- No differential model of the environment (including the pursuers) is available, as the robot platforms for the evader and pursuers are unknown.
- The sensed number of pursuers in the local environment can be less than the total number of pursuers in the global environment.

A solution to this kind of game, where the evader does not know the strategy followed by pursuers, nor the entirety of the environment, is the computation of a strategy (set of navigation steps) that prevents capture as long as possible. Preventing capture is done in terms of increasing the distance between the pursuers and the robot (evader), or by using possible hideaways. The strategy is computed from simulations, in the sense that the process of trial and error of actions is not performed in reality. The robot computes imaginary interactions with pursuers, from a designated model (e.g., worst-case scenario), and when the strategy is ready, it can be implemented in reality to achieve the planned motions.

7.2 Solving the game using reinforcement learning

Reinforcement learning has been chosen for the computation of strategies as a solution to the problem of navigation and escape due to the following advantages, particular to the Q-learning algorithm:

- Reinforcement learning is unsupervised, as the learning is performed from trial and error of actions and the evaluation of the consequent states.
- Reinforcement learning does not require a differential model of the environment and its elements, but only a transition kernel (that contains the transition probabilities and the reward function).
- Q-learning and other popular reinforcement learning algorithms have been used in autonomous navigation learning tasks and pursuit-evasion games previously, but focusing on pursuit strategies.

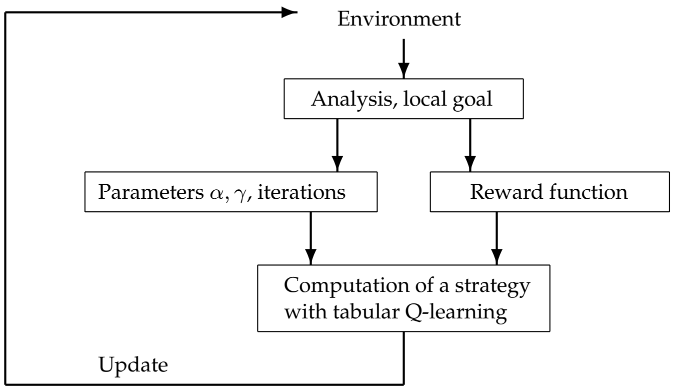

Environment
Analysis, local goal
Parameters α, γ, iterations
Reward function
Computation of a strategy with tabular Q-learning
Update

Figure 7.1: Schematic of the elements in the proposed designs for computing navigation strategies with reinforcement learning.

- The convergence of an MDP Q-learning algorithm for one agent has been proved, so optimal strategies can be computed within the correct set of learning parameters.

The proposed use of reinforcement learning follows the decisions regarding the control strategy and solutions to the problem described in Section 4.2. A strategy is a high level control system, consisting of a series of discrete navigation targets. The computed strategies need to be updated as the environment changes, when the robot observes different parts of it, and the pursuers react to the actions of the robot (sometimes in a different way than predicted). The robot computes a new simulated interaction with the pursuers, and consequently a new strategy that considers the recent information and the true location of the pursuers. The environment representation, states and allowed actions are considered discrete.

The solutions based on reinforcement learning proposed in this chapter are divided as follows for their design, according to the methodology by Hwang and Ahuja (1992a), and illustrated in Figure 7.1:

1. A biologically inspired analysis of the environment that uses the same structure as the one presented in Chapter 4, to identify the elements from the sensing, and then computing a local goal and the relevant reinforcement learning parameters.
2. The particularities of the Q-learning algorithm that learns the navigation strategy: selecting features like the action selection mechanism, shaped reward functions and other learning parameters.

A realistic implementation of autonomous navigation involves a need for short-term actions (Luo et al., 2008), thus the learning process should be fast enough to provide a strategy for navigation. The next subsections describe the design of the analysis of the environment and the selection of parameters for a Q-learning algorithm. The selection of the features and parameters in a Q-learning algorithm dictates the speed of the computation of a strategy, and helps to find a global optimal strategy within the available possibilities if enough exploration is provided, thus forming an important part of the design of the learning.

7.2.1 Biologically inspired analysis of the environment

A cell-decomposition representation of the environment into an occupancy grid, like the one proposed in Chapter 4, has been considered to map the identified elements. The best location for the robot in a sensed environment is calculated in terms of the distances to all the pursuers, and from the concept of hideaways. The procedure and equations in Section 4.4 are the same:

1. Approximating the environment into a grid, where elements are identified and approximated to cells according to the dimensions of the robot, and independent probabilities are assigned according to the accuracy of the information: $P_{(a,b)}(o)$ if a cell is a probable obstacle, $P_{(a,b)}(p)$ if a cell is a probable pursuer, and $P_{(a,b)}(F_s)$ if the cell is probable free space.
2. Computing the possible hideaways (bio-inspired refuges) as the shadows considering all the cells in the environment, $w_{hideaway}$.
3. Computing a measure of the distances between any possible free space cells and the pursuers and the position of the robot, $w_{distances}$.
4. Computing a total function in terms of the distances and the hideaways that allows choosing a local navigation goal, LG.

Some of the functions computed in the analysis of the environment have been used for the features in the Q-learning algorithm, like the measure of the distances in step 3 and the local goal location for the reward function. The selection of features and parameters, and the shaping of the reward function are presented in the following subsection.

7.2.2 Parameters for the Q-learning algorithm

The tabular Q-learning algorithm, shown in the previous chapter (Algorithm 3), has been selected for the learning of the navigation strategy (or policy) π^*, as the states and actions are discretised in a grid form. The main features and parameters that need to be determined for Q-learning include: the action selection mechanism and its own parameters; the learning rate (α), the discount factor for the rewards (γ), the total number of iterations or episodes (from the initial state to the terminal), the rate of decrease of the parameters and the reward function. The selection of all these features and parameters affect the balance between exploration and exploitation of actions to find an optimal policy, and the speed of the computations, as mentioned previously.

Action selection mechanism

An ϵ-greedy action selection policy (Sutton and Barto, 1998) was selected for the computations, as it allows a balance between exploration and exploitation. As mentioned in Section 6.4.4, an action (from the allowed actions for a state, $A = \mathcal{A}(s)$) is selected with the following probabilities:

$$P(a) = \begin{cases} 1-\epsilon & \text{if } a = a^* \\ \frac{\epsilon}{A} & \text{otherwise} \end{cases} \tag{7.1}$$

The exploration is decreased in a linear form, like $\frac{\epsilon}{iter}$, according to the number of episodes (or iterations) of the learning, to shift towards exploitation of the optimal values (Even-Dar and Mansour, 2003).

The action selection mechanism considers the allowed actions for a state, $\mathcal{A}(s)$, as those leading to free space and not towards collision with obstacles or pursuers. The actions that would produce collisions are pruned from the set $\mathcal{A}(s)$ before assigning the probabilities, ϵ.

Dynamic learning parameters

The process of learning a navigation strategy can be very slow depending on the parameters of the algorithm. In the task of navigation and escape from pursuers, a faster learning is needed according to a major threat by pursuers. Therefore, the parameters for the learning need to be adjusted according to the demands of the task. Dynamic selection of parameters is proposed from a measure denominated *global danger*.

A measure of global danger, D, has been defined in terms of the available information from the sensing, corresponding to the distances or relative locations of the pursuers within range. The number and speed of pursuers should determine how urgent the computation of a strategy becomes. For a total of n_p pursuers,

$$D = \sum_{p=1}^{n_p} \frac{danger_p}{n_p} \tag{7.2}$$

where the individual contribution to the danger by a pursuer, $danger_p$, is a function of the distances between the p pursuer and the robot,

$$danger_p = \begin{cases} \frac{1}{d(p,\text{robot})} & \text{if } d(p,\text{robot}) > 0 \\ 1 & \text{if } d(p,\text{robot}) = 0 \end{cases} \tag{7.3}$$

where $d(p,\text{robot})$ is the Euclidean distance between a pursuer and the robot. When D is close to a value of 1, the pursuers are in adjacent cells to the robot and the danger is maximum. Thus an action is needed immediately and any deliberation demands a fast computation.

The use of fixed sets of parameters, chosen from the resulting global danger, D, is proposed. The sets of parameters have been selected from experimentation, as presented in Subsection 7.2.3, considering the need for convergence to a usable navigation solution after the learning, regardless of the speed in the learning. The initial parameters are decreased or increased in a linear form, like $\gamma + \frac{\gamma}{20 \cdot iter}$, to shift the learning from exploration to exploitation as the number of iterations increases (Even-Dar and Mansour, 2003).

Table 7.1: Table of rewards (Reward 1) for actions and resulting states.

FEATURE	REWARD OR PUNISHMENT
Collision with an obstacle or pursuer, by reaching a cell occupied by an obstacle or pursuer	-1000
Reaching the local goal cell	1000
Navigating to a cell that is closer to the local goal	10
Staying in the same cell before reaching the local goal	0
Navigating to a cell that is further away from the local goal	-10
Reaching the limits of the grid without reaching the local goal	0

Shaping the reward function

Some authors have proposed incorporating relevant information in the reward function to facilitate and accelerate the learning (Matarić, 1994). With this purpose in mind, a first idea contemplates the use of simple responses to the presence of obstacles and pursuers in the environment and the motion towards a local goal. Constant rewards and punishments are assigned to the actions according to the state transition as shown in Table 7.1.

A second proposal is the combination of information from the elements in the environment (the presence of obstacles and pursuers, and the distances), similar to a potential function. The elements in the reward function include:

- A reward is necessary to guide the robot away from all pursuers simultaneously.
- A punishment should be given if the robot gets too close to obstacles or pursuers.
- If a local goal is available, the reward function should teach the robot to reach it by reinforcing the motion towards it.

The proposed shaped reward function is the result of weighted components of the elements,

$$R_t = R_c + R_{LG} \tag{7.4}$$

where R_c penalises the closeness of the robot to any obstacle or pursuer to prevent collisions, and R_{LG} rewards the distance towards the local goal. The penalisation against collisions is defined as

$$R_c = \begin{cases} -1000 & \text{if } d(\text{robot, pursuer or obstacle cell}) \leq d_c \\ 0 & \text{otherwise} \end{cases} \tag{7.5}$$

if the robot is in a circular region from the centre of any obstacle or pursuer cell, with radius d_c and a distance between the robot and the cell of $d(\text{robot, pursuer or obstacle cell})$. The reward for the distance towards the local goal is the inverse quadratic attractor potential function (Subsection 4.5.1),

$$R_{LG} = \frac{2}{K_{LG} \cdot d((a,b), LG)^2} \tag{7.6}$$

where $d((a,b), LG)$ is the distance between a cell and the chosen local goal, and K_{LG} regulates the magnitude of the attraction. The selection of the local goal proposed in

Chapter 4 depends on a function of the distances to all the probable pursuers, as well as the biologically inspired concept of refuges in the form of hideaways behind obstacles that block the line of sight of pursuers. Thus, a navigation towards the local goal (which is updated according to the sensing of new elements in the environment and the new positions of the pursuers) would lead to a location away from all the pursuers simultaneously, and maybe reaching a hideaway.

Improving the policy

The strategy for the navigation towards the local goal, i.e. away from pursuers and possibly towards a hideaway, is computed through the resulting action-value function, $Q(s, a)$, from the iterative learning process, as explained earlier. For each state s, the action that maximises the value of $Q(s, a)$ is the optimal policy,

$$\pi^*(s) = \max_{a \in \mathcal{A}(s)} Q^\pi(s, a) \tag{7.7}$$

Updating the navigation and escape strategy

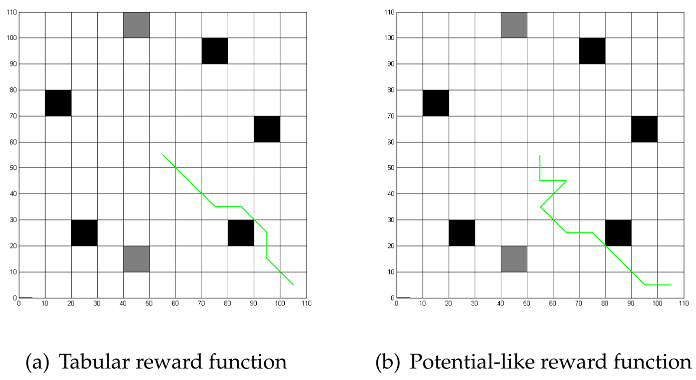
When the environment changes due to the observation of new elements from the new location of the robot, the last choice of local goal could need to be updated along with the navigation strategy. Thus, the learning process is repeated to compute a newer strategy, as many times as needed, and according to the available computational resources. This process is implemented in an equivalent manner to the updates for the path planning in Subsection 4.4.6.

7.2.3 Empirical selection of ranges of operation of the parameters

The main objective of the simulations presented in this subsection is to identify the ranges of operation for the learning parameters in the Q-learning algorithm, considering the proposed reward functions. The identification of the parameters that correspond to a desired performance of the learning provided fixed sets according to the needs of the task, in terms of the measure of global danger D introduced before.

The simulations consisted of evaluating the root-mean-square error (RMSE) and the total deliberation time of the learned navigation strategy towards the local goal while one of the parameters is varied and the other fixed, comparing that learned strategy with an optimal strategy. This process is repeated for all the strategies, and the average of the RMSE and the deliberation time is computed. The RMSE is explained in Appendix A. The optimal strategies for comparison were computed starting in the centre of the grid, for both reward functions proposed in Subsection 7.2.2, and is shown in Figure 7.2 (Sutton and Barto, 1998).

The parameters used to compute the optimal strategies are: 10000 iterations, allowing exploration with $\epsilon = 0.5$, $\alpha = 0.1$, $\gamma = 0.6$, and linear decreasing or increasing from exploration to exploitation. It is possible to observe that for the tabular reward function, the



(a) Tabular reward function (b) Potential-like reward function

Figure 7.2: Optimal strategies for the two proposed reward functions and a fixed environment: the tabular reward function and the potential-like reward function.

Table 7.2: Evaluation of the effect of the parameters in the computation of a strategy.

PARAMETER AND VARIATION	SET 1	SET 2	SET 3
Learning rate α $[0.1, 0.9]$ at intervals of 0.1	0.5	0.3	0.1
Discount rate γ $[0.6, 0.9]$ at intervals of 0.1	0.999	0.8	0.6
Exploration probability ϵ $[0, 0.5]$ at intervals of 0.1	0.1	0.25	0.5
Number of iterations $\{50, 100, 500, 1000, 5000\}$	500	1000	5000

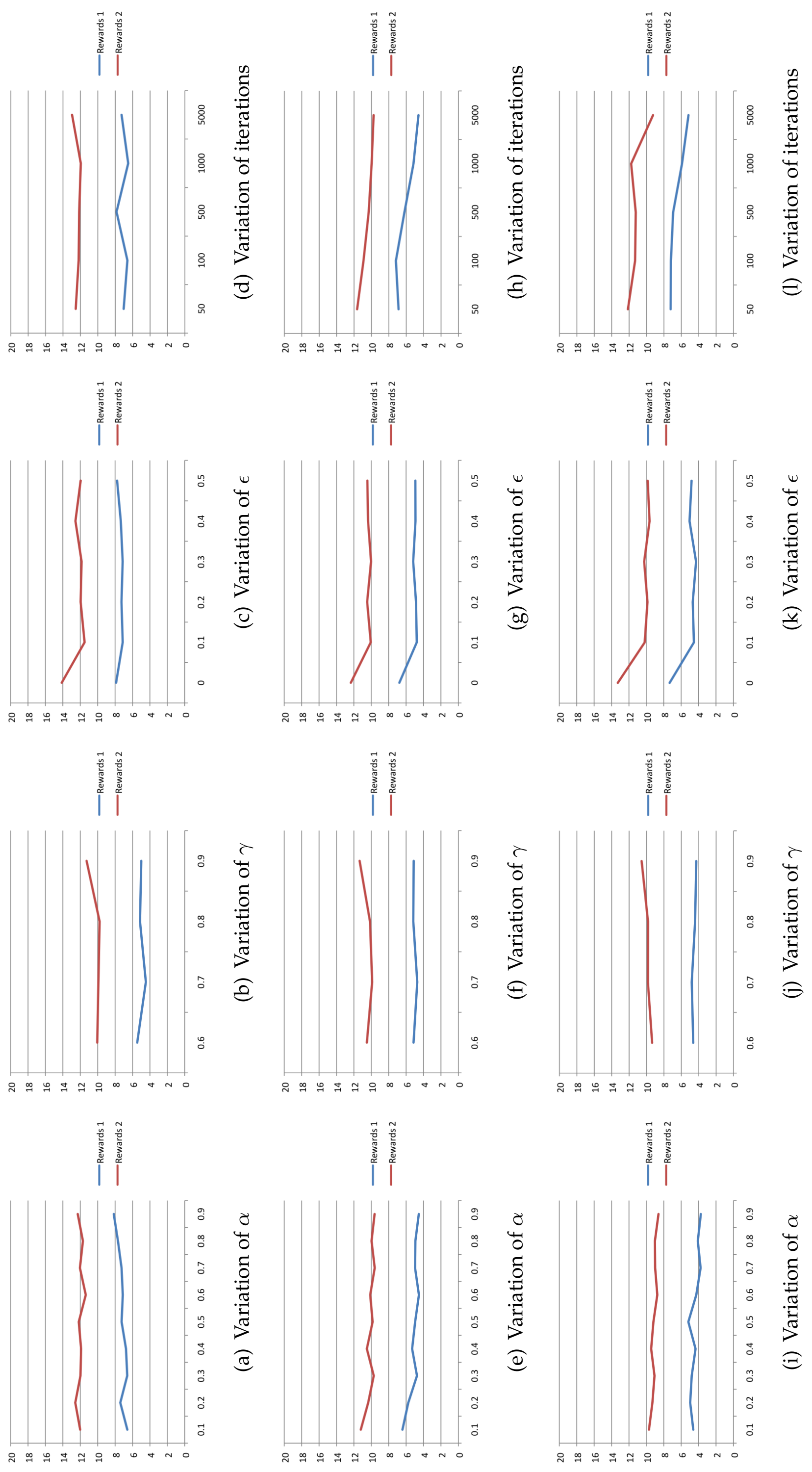
optimal navigation strategy is a direct route to the local goal, whereas the second path leads to less efficient navigation in terms of the shortest distances to the goal. The convergence properties of tabular Q-learning allow the assumption that the resulting path, after all the iterations, is close to the optimal.

The settings for the simulations are:

- A discrete, randomly generated 2-D environment, consisting of a grid of 11×11 cells, 5 obstacles, and 2 pursuers.
- 100 repetitions of each experiment, keeping the same parameters (varied and fixed), to compute the average of the RMSE and the average of the deliberation time.
- For the potential-like reward function, $d_c = 1.5$ cells, $K_{LG} = 0.001$ and R_{LG} has been limited to a maximum of 1000.
- Variation of one parameter at a time, while the other were fixed in one of the sets shown in Table 7.2.

Figures 7.3 and 7.4 show the average of the RMSE and the average of the deliberation time, according to the variation of parameters for the tabular reward function and the potential-like, respectively. Reward 1 corresponds to the tabular reward function, whereas Reward 2 to the potential-like function (R_t).

The parameters in set 1 corresponds to the exploitation of the optimal values in the learning process and the computation of the strategies, and to a fast computation of the results in less iterations. The set 3 corresponds to the initial random exploration of many state-action pairs, and experimentation through a large number of iterations, before computing the



(a) Variation of α
(b) Variation of γ
(c) Variation of ϵ
(d) Variation of iterations
(e) Variation of α
(f) Variation of γ
(g) Variation of ϵ
(h) Variation of iterations
(i) Variation of α
(j) Variation of γ
(k) Variation of ϵ
(l) Variation of iterations

Figure 7.3: Average of the RMSE respect to the optimal solution according to the variation of the learning parameters (α, γ, ϵ and the number of iterations), using the sets 1 to 3 (first to last rows) as fixed. The horizontal axis corresponds to the value of the variable, and the vertical axis to the average of the RMSE.

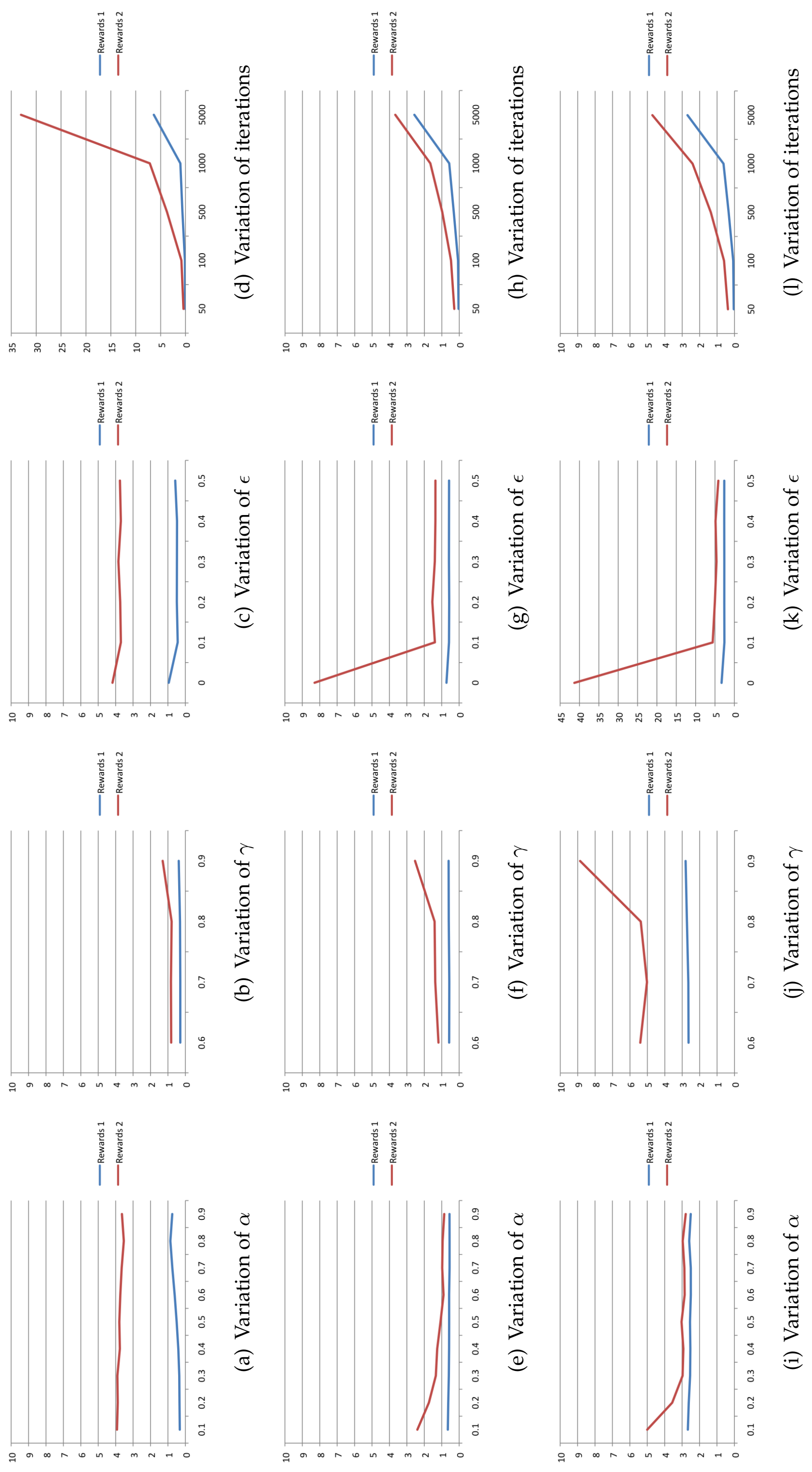


Figure 7.4: Average of deliberation time according to the variation of the learning parameters (α, γ, ϵ and the number of iterations), using the sets 1 to 3 (first to last rows) as fixed. The horizontal axis corresponds to the value of the variable, and the vertical axis to the average of the deliberation time in seconds.

Table 7.3: Parameters according to the value of global danger, D.

PARAMETER	$D < 0.2$	$D \geq 0.2$
Learning rate α	0.5	0.6
Discount rate γ	0.7	0.6
Exploration probability ϵ	0.5	0.3
Number of iterations	100	50

optimal strategy after a long time. The set 2 represents a balance between exploration and exploitation, by using parameters between the specifications of sets 1 and 3.

The results in Figure 7.3 show that the RMSE decreases considerably when more exploration and a larger number of simulations take place, for both kinds of reward functions. The average RMSE for the simulations with parameters from set 2 is intermediate to the results from sets 1 and 3. It has been noted, from both figures, that the results for the tabular reward function (Reward 1) improve both the RMSE and the deliberation time of the potential-like function.

Extreme peaks in the average deliberation time are observed for ϵ and the number of iterations, in Figure 7.4. The peak for $\epsilon = 0$ occurs when the other parameters are set for a slower learning and with exploration (set 3), whereas the peak for the number of iterations (of 5000) occurs when the fixed parameters are set for exploitation (set1). This indicates that the selection of parameters needs to be consistent with the expectations of exploitation of exploration in the learning. Furthermore, a balance between exploration and exploitation through the selection of appropriate learning parameters leads to adequate results in the convergence to an optimal strategy, and at the same time, within a reasonable time.

The results in Figure 7.4 of the average deliberation time were mainly considered to propose two different sets of parameters according to a threshold of danger, $D = 0.2$, where if the danger is less, more exploration takes place. But if the danger is more, priority is given to exploitation, considering some exploration to achieve a good solution. The selected sets of parameters for both sides of the threshold are shown in Table 7.3.

Another proposed option to compute dynamic parameters is the use of functions limited to a selected range of operation. In related publications, similar simulations to evaluate the ranges of operation with respect to the visits to all state-action pairs, and the total deliberation time were computed (Araiza-Illan and Dodd, 2010). Useful ranges of operation for the learning parameters (α, γ, ϵ, and the number of iterations) were identified, and exponential functions proposed in terms of the same measure of D, from Subsection 7.2.2.

7.3 Simulations

The simulations presented in this section have the following objectives:

1. Evaluating if the proposed reinforcement learning approaches (the computed navigation strategies) prevent the capture of a robot in the window of time of the test

(20 navigation steps), for different velocities of pursuers looking to capture greedily.

2. Analysing which one of the reward functions prevents the capture for the different numbers of pursuers and velocities.
3. Evaluating the effect of the updates of the sensing (measured in navigation steps) in the prevention of capture by pursuers with different velocities.

The success of a path occurs when the robot avoids capture in 20 navigation steps, or if capture happens later (in navigation steps) than with the other path corresponding to the other reward function.

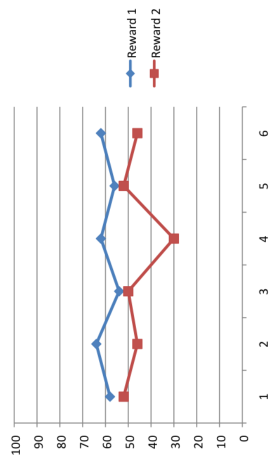
The parameters for the simulations in this section are:

- 50 randomly generated environments.
- Low resolution grid of 11×11 cells.
- The sensed obstacles and pursuers are not extended to adjacent cells.
- 1 to 5 and 10 pursuers, with velocities of $v_p = v_r, 2v_r, \frac{1}{2}v_r$.
- The sensing is updated every 1, 2, 3, 4, 5 and 6 navigation steps, equivalent to a cell per step.
- The robot starts an experiment in the centre of the grid, and it moves from cell to cell.
- Both reward functions, the one from Table 7.1, and the potential-like. For the potential-like reward function, $d_c = 1.5$ cells, $K_{LG} = 1$,and R_{LG} has been limited to a maximum of 1000.
- The tabular reward function from
- The learning parameters used in the Q-learning algorithm, as shown in Table 7.3, from the value of D.

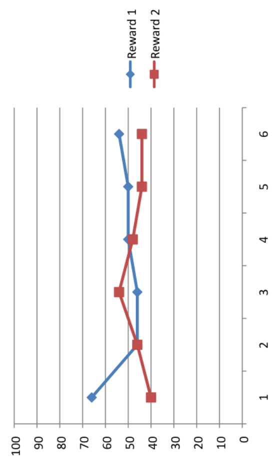
The flow of the computations for the simulations in this section is the same as Figure 7.1.

Significant results are presented in the form of comparative graphs, according to numbers of pursuers, and their velocities. Figure 7.5 shows the results on the success for 1, 3 and 10 pursuers, with updates every 1 to 6 navigation steps. Also, the average of the time of capture in navigation steps, for the same numbers of pursuers, are shown in Figure 7.6, according to the Other graphs for 2, 4 and 5 pursuers are shown in Appendix E. The legends correspond to the two different reward functions, as in the previous simulations in this chapter.

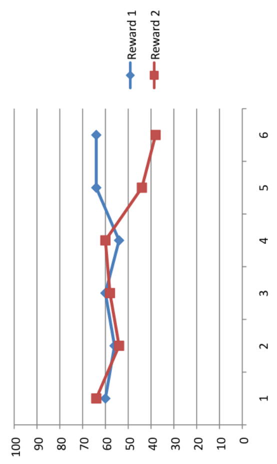
The results indicate that for few pursuers (1 pursuer) or many (4, 5 and 10 pursuers), both reward functions perform similarly in terms of the computed measure of success. For 2 and 3 pursuers, the tabular reward function performs better comparatively than the potential-like reward function. Also, when the pursuers are slower or fewer (1 pursuer), updating the sensing in the environment more frequently provides the best results. For faster and more pursuers, the update of the sensing does not make a big impact in the success, nor the steps before capture.



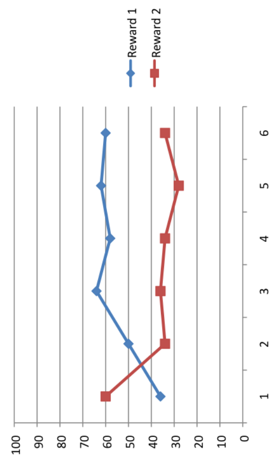
(a) 1 pursuer, $v_p = v_r$



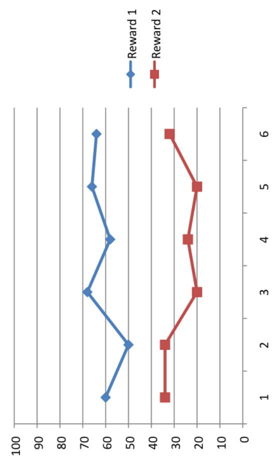
(b) 1 pursuer, $v_p = 2v_r$



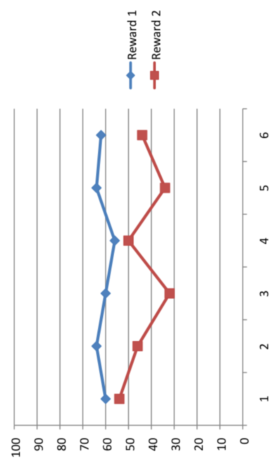
(c) 1 pursuer, $v_p = \frac{1}{2}v_r$



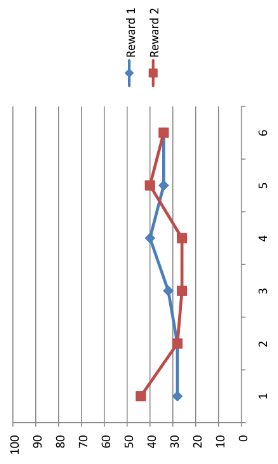
(d) 3 pursuers, $v_p = v_r$



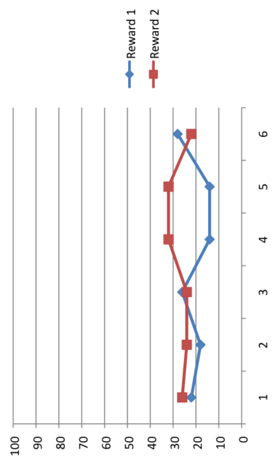
(e) 3 pursuers, $v_p = 2v_r$



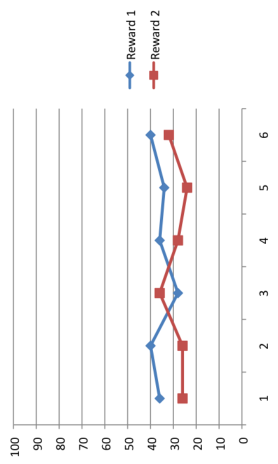
(f) 3 pursuers, $v_p = \frac{1}{2}v_r$



(g) 10 pursuers, $v_p = v_r$

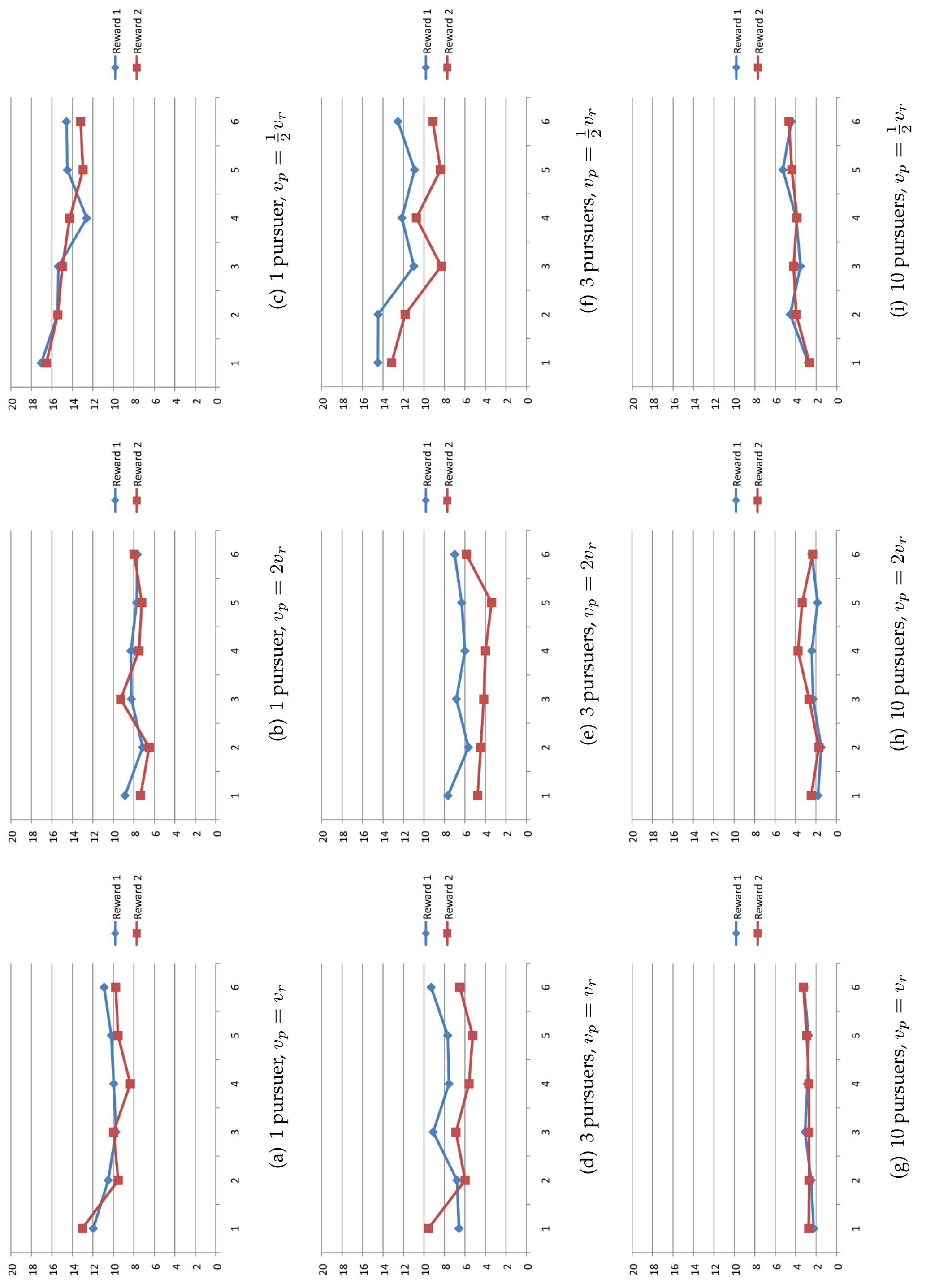


(h) 10 pursuers, $v_p = 2v_r$



(i) 10 pursuers, $v_p = \frac{1}{2}v_r$

Figure 7.5: Success of the strategies for 1, 3 and 10 pursuers with different velocities. The horizontal axis corresponds to the sensing updates (in navigation steps) and the vertical axis is the percentage of success.



(a) 1 pursuer, $v_p = v_r$

(b) 1 pursuer, $v_p = 2v_r$

(c) 1 pursuer, $v_p = \frac{1}{2}v_r$

(d) 3 pursuers, $v_p = v_r$

(e) 3 pursuers, $v_p = 2v_r$

(f) 3 pursuers, $v_p = \frac{1}{2}v_r$

(g) 10 pursuers, $v_p = v_r$

(h) 10 pursuers, $v_p = 2v_r$

(i) 10 pursuers, $v_p = \frac{1}{2}v_r$

Figure 7.6: Capture times (in terms of navigation steps), for 1, 3 and 10 pursuers with different velocities. The horizontal axis corresponds to the sensing updates (in navigation steps) and the vertical axis is the average of capture time (also in navigation steps), for 50 simulations.

7.4 Discussion

Reinforcement learning was chosen for the designs presented in this chapter due to its previous use for solutions to the problem of general autonomous navigation, as well as its use for solutions to pursuit-evasion games, but from a pursuit perspective. The main advantage of reinforcement learning for the particularities of the problem is that no mathematical model of the environment is required to compute a strategy, but only rewards from the transition of the states after selecting actions. Nevertheless, knowledge about the possible actions, the transitions, and states are needed. This knowledge is provided by the analysis of the environment from sensed local information, proposed in Chapter 4.

The proposed implementation of the learning is based on simulating the interaction with the pursuers and the environment to compute a navigation strategy, before the real one takes place. The same simulating process used to evaluate the path planning approach has been adapted to be coupled with the reinforcement learning to compute a navigation strategy. The biologically inspired environment analysis computes the local navigation goal, a representation of the current state and the possible navigation actions accordingly. The global structure repeats the same process as the path planning proposals, but exchanging *a priori* proteanism plus steepest descent search for a tabular Q-learning process to compute a navigation strategy.

Differences between the tabular Q-learning approach and others from the literature include the dynamic selection of learning parameters, based on an assessment of the identified pursuers in the environment. The learning parameters (α, γ, ϵ and the number of iterations) are chosen according to the presence of pursuers, to favour a fast computation of the optimal strategy that also balances exploration and exploitation. Favouring a faster deliberation time, the proposed shaping of the reward functions incorporates the pursuers, as well as the local goal computed previously. The results from the simulations to select appropriate learning parameters with the two proposed reward functions show that for this task of navigation and escape, a good balance between exploration and exploitation is needed to compute reasonable strategies in a small amount of time.

The simulations to evaluate the performance of the proposed Q-learning based approach when interacting with pursuers in randomly generated environments, are of similar characteristics to the ones in Chapter 5. The results of the evaluation of success in avoiding capture show that the reinforcement learning approach does not perform as well as the path planning with potential functions for similar simulations (with the same randomly distributed environments and sensing) in low resolution grids, pursuers of different speeds and updating the environment sensing. This situation is more evident in the results on the average time before capture. Comparing Figures 5.21, 5.22 and 5.23 from Chapter 5 with 7.6 from this chapter, the later are equal or lower and thus the success of the strategies from reinforcement learning is lower. Also, reinforcement learning based navigation strategies are more computationally expensive than deliberating a path, and consume more time in their computation.

7.5 Concluding remarks

This chapter started by presenting an extension of the problem of navigation and escape from pursuers from Section4.1, to the pursuit-evasion games domain, for its solution through reinforcement learning algorithms. Then, the main components of the designs are presented:

- A biologically inspired analysis of the environment to compute a local navigation goal, based on the distances to pursuers and possible hideaways, as presented previously in Chapter 4.
- The selection of parameters for a tabular Q-learning algorithm, including: the action selection mechanism, the learning parameters, the reward function, the policy improving method, and the updates of the deliberation process according to the changes in the environment. The values for the learning parameters were chosen from experimentation, through evaluating the average RMSE and the average deliberation time when varying one parameter at a time.

Subsequently, simulations to evaluate the success of the computed navigation strategies were presented. The effects of the different reward functions and update of the sensing served to evaluate the performance of the generated strategies avoiding capture for longer. The results are presented in the form of comparative charts.

The discussion of the results highlights the main features of the presented designs, as well as the findings through the simulations. The proposed reinforcement learning approach is briefly compared to the path planning approach presented in Chapter 4 and evaluated in Chapter 5, in terms of the success. Results show that the path planning approach with potential functions is better in terms of navigation time before capture compared to the reinforcement learning approach. Nevertheless, the reinforcement learning algorithms are easier to program, apply and understand. Also, the simple tabular reward function (Reward 1) resulted more successful compared to the potential-like function (Reward 2).

The next chapter presents a description of the implementation of some of the proposed algorithms in a real robotic platform, the e-Puck. The sensing and low level control system used to couple the high level planning with the robot are detailed, along with some visual examples of the results.

Chapter 8

Implementation on real robots

This chapter demonstrates the implementation of the algorithms proposed in Chapter 4 on a robotic platform. The work was done in collaboration with Charles Winter, under the SURE research program in the Department of Automatic Control and Systems Engineering.

The e-Puck robot has been chosen to represent both the robot and the pursuers, and a small environment was built to evaluate the performance of the implementation. Section 8.1 presents the characteristics of the used robotic platform. The main features of the e-Puck are described, highlighting its comparative advantages over other commercially available robots.

Section 8.2 describes the global control system implemented in the robot. The control system has been divided into the stages of sensing, deliberating and moving. The procedure for the sensing using the camera embedded in the robot, the parameters for the planning of the motion according to the proposals in Chapter 4, and the implementation of the real movement are presented.

Section 8.3 shows some examples of the performance of the robot with one pursuer and the environment with obstacles. Subsection 8.3.1 describes the implementation of the control system for the navigation of the pursuers, and Subsection 8.3.2 shows the results through images of the motion.

Finally, Section 8.4 presents the discussion of the results, highlighting the main findings on the implementation of the sensing and low level motion control. Section 8.5 presents the concluding remarks of the chapter.

8.1 Physical characteristics of the e-Puck robot

The main characteristics of the e-Puck, relevant to the implementation presented in this chapter are (Mondada et al., 2009):

- Size: a diameter of 75 mm.

- Motion: 2 motors with PID control integrated.
- Available long distance sensors: an integrated camera, with a resolution of 40×40 pixels at 4 frames per second. Odometer for the position in the plane.
- Communication: Bluetooth module integrated, for the communication with MATLAB running in a PC to process the sensing and to perform the planning of a navigation path.
- Programming: the micro-controller can be programmed in C or assembly language through MPLAB, to perform the sensing, communicate with MATLAB and then execute the motion.

The main advantages for the use of this robotic platform include:

- The robots come pre-assembled from the factory, sensors included.
- Extensive documentation for the programming of the robot is available online.
- Low level libraries are available for the control of the camera and motors.
- The protocol for the Bluetooth connection with a computer is integrated in the software.
- The control of the motors is performed by indicating the speeds and time according to the difference between the desired position and the odometer information.
- The size of the robot allows performing simulations in reduced spaces.

On the other hand, the main disadvantages are:

- The price per unit without adding accessories.
- The limitations of the sensing, as the camera only covers an angle of 53 degrees. Thus, the sensing of the elements behind and to the sides of the robot involves a discontinuity in a linear motion and a delay in time.

8.2 Global control system

The robot obeys a global process to solve the problem of autonomous navigation and escape from pursuers, divided in three stages: sensing, deliberating the navigation plan and moving accordingly. The process is repeated as the observed regions of the environment change due to the motion of the robot. The stages of the sensing and moving involve an interaction between the high level control system that performs the planning, and the low level control in charge of the interaction with the hardware (camera and motors). Figure 8.1 shows the process as a flow diagram.

The following subsections present the three stages, Subsection 8.2.1 the sensing of the elements, Subsection 8.2.2 the planning of the motion, and Subsection 8.2.3 the motion following the path, correspondingly.

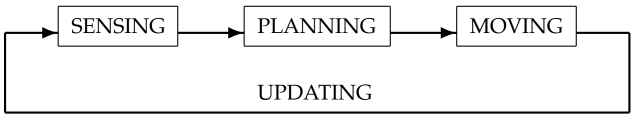
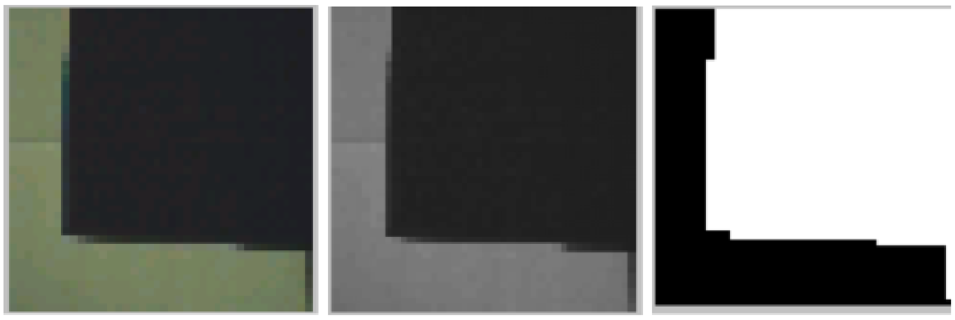


Figure 8.1: Flow diagram of the global control system.



(a) RGB (b) Greyscale (c) Binary

Figure 8.2: Transformation of an image from RGB to greyscale and binary formats.

8.2.1 Sensing the elements in the environment

The robot takes a total of 8 colour snapshots of its surroundings with a maximum reach of 3 times the size of the robot, approximately, to have a complete range of vision. The images are transferred to MATLAB, to execute an analysis that generates an occupancy grid of the elements in the local observed environment. The environment around the robot is assumed to consist of dark blue obstacles, red pursuers, and white floor and walls that limit the space.

The analysis of the images from the camera consists of the following steps for each image:

1. Analysing the brightness and colour composition of an image. The image is mapped from a RGB format to the format $L_b \cdot c_1 \cdot c_2$, where L_b indicates the level of brightness of the image; c_1 is a value from -10 to 10, with -10 being a fully green image, and 10 being a fully red image; and c_2 is a value from -10 to 10 on a scale of yellow to blue. The resulting value of c_1 from an image is compared with a predetermined threshold. If the threshold is exceeded, this indicates the presence of a pursuer.

   The distance between the pursuer and a robot is determined from the brightness of the image, L_b. The brightness is significantly higher if there is a distance between the pursuer and the robot, due to the whiteness of the floor. Empirically calculated thresholds for the brightness has been used to determine the approximate values of the distances.

2. The image is transformed into a greyscale format, and then into a binary image (black and white), as shown in Figure 8.2. The dark pixels in the image are assigned a value of zero, and light colour pixels a value of 1. If the resulting average value for all the pixels in the image is high enough, an obstacle is present.

The distance between the obstacle and the robot is determined from the transposition of the image and the average of black and white of all the columns of pixels. The number of columns from left to right, with a high average, correspond to the amount of white above the object, and it can be compared to empirical measures to determine the value of the distance.

3. When no obstacles or pursuers are identified in an image, it indicates the presence of free space only. This can be computed analytically by comparing the average value for all the pixels in the binary image, as the value should be low due to the whiteness of the floor.

The computed distances from all the images are mapped into a grid of 6×6 cells, indicating the presence of obstacles, pursuers, free space or unknown space. The occupancy grid is used by the planning algorithm to deliberate a navigation path. The set of probabilities for each cell is considered to be hard (1 for the corresponding detected element), according to the presence of obstacles, pursuers and free space.

8.2.2 Planning algorithm

The planning process to compute a navigation path has been described in Chapter 4. The following general steps are executed, from the occupancy grid map computed in the sensing:

1. Computing the function of the distances w_d, the function of the intersection of shadows, w_h, and selecting a local goal accordingly.
2. Computing guided subgoals towards the local goal, for *a priori* proteanism.
3. Building the potential functions, from modified FIRAS and quadratic goal, according to the pursuers, obstacles and subgoals.
4. Deliberating the path from steepest descent with distances, by commuting the subgoals in the potential functions.

The path towards the local goal (except if local minima problems appear in the potential functions) is used to guide the robot until the new sensing takes place and a newer path is computed.

8.2.3 Low level control for the movement

The path deliberated by the planning algorithm is produced in terms of cartesian coordinates. A transformation into polar coordinates (angle and displacement) takes place before they are sent back to the robot, via Bluetooth, to be implemented in the low level control.

The motion from one cell to the next is executed by two different motions:

1. First, turning to the right angle. A proportional control decides the amount of turning needed to correct the angle of the robot, θ_r, by calculating the time a fixed speed in

the wheels is applied to execute the rotation. A speed of $[1000, -1000]$ for the motors on the left and right, respectively, is needed for a full speed, clockwise turn around the axis at the geometrical centroid of the robot.

2. Then, executing the displacement towards the centroid of the target cell. A maximum speed of $[1000, 1000]$ is sent to the two motors (left and right) for a fixed length of time, to achieve the displacement of one cell in the grid at a time. The time is calculated from the difference between the odometer reading and the desired location.

The following section shows some examples where the motion of the robot is executed according to the sensed elements in the environment.

8.3 Examples of the navigation of the robot

In order to verify the correct performance of the proposed global control system according to the simulations, some experiments were recorded and the results are presented visually in this section. An autonomous pursuer that follows the robot around was programmed to complement the interactions with the environment.

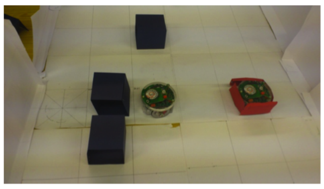
8.3.1 Programming the pursuer

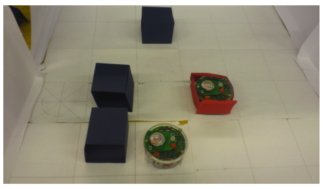
The pursuer receives the current location of the robot at all times, but it will only pursue if the robot is within sensing distance (3 cells, same as the robot). The objective of the pursuer is to follow the robot, by moving to a location within the adjacent cells that minimises the distance to the robot. The pursuer is able to avoid collisions by eliminating the unavailable adjacent cells from the minimisation of the distance process, and when the robot is invisible as it is behind an obstacle, the pursuer does not move. The motion is executed following the same low level control for the robot, by turning first and then displacing towards the corresponding cell.

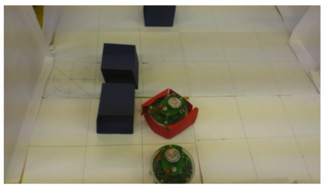
8.3.2 Examples

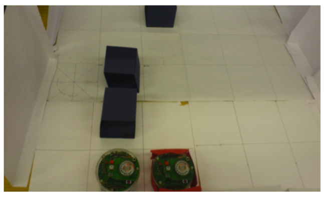
In the simulations, a path based on guided subgoals is computed when the pursuer is detected, with a radius for the subgoals of $r_{sg} = 2$ and using modified FIRAS and quadratic goal potential functions. Other parameters include $K_g = 0.1$, $K_i = 1$, $d_o = 1$cell, and a maximum potential value of 1000. The pursuer has the same velocity as the robot. The environment consists of a surface of 6×6 cells, surrounded by walls, with scattered obstacles of a single cell size.

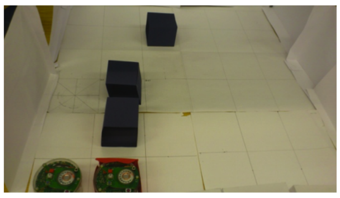
The first example, Figure 8.3, shows the resulting path when the sensing is updated every 4 navigation steps, and the second example, Figure 8.4 for updates every 2 navigation steps. The examples, and some other simulations, show that performing updates of the sensing closer in intervals of time led to better paths to avoid capture for longer.


(a) First motion


(b) Second motion


(c) Third motion


(d) Fourth motion


(e) Fifth motion

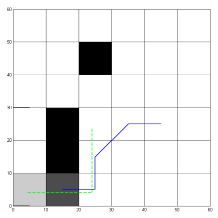
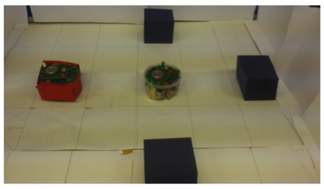
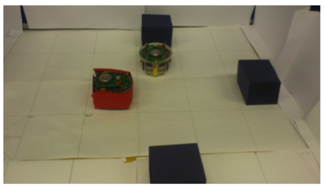
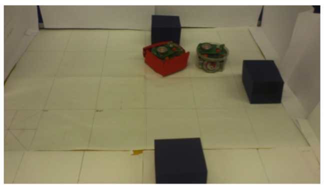
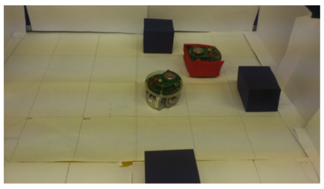

(f) Global paths

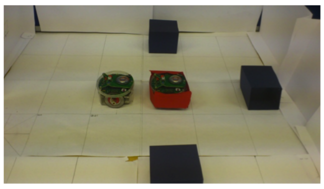
Figure 8.3: Example of the motion of a robot in an enclosed environment, for $v_p = v_r$, and the global followed paths in the grid; path of the robot in green, path of the pursuer in blue, with their final positions indicated by the squares (light grey is the robot, and dark grey is the pursuer); obstacles in black.

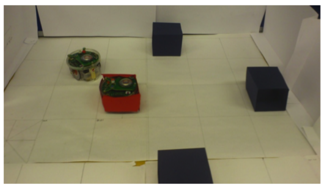

(a) First motion

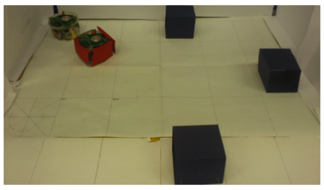

(b) Second motion


(c) Third motion


(d) Fourth motion


(e) Fifth motion


(f) Sixth motion


(g) Seventh motion

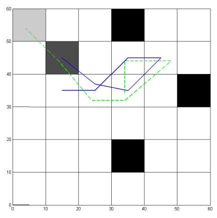

(h) Global paths

Figure 8.4: Example of the motion of a robot in an enclosed environment and global followed paths in the grid; path of the robot in green, path of the pursuer in blue, with their final positions indicated by the squares (light grey is the robot, and dark grey is the pursuer); obstacles in black.

8.4 Discussion

The simulations of the interactions with pursuers and obstacles demonstrate that the planning algorithm works in the implementation presented in this chapter. The robot moves towards locations away from pursuers and around or behind obstacles according to the sensing. Closer updates of the sensing produce better results for few pursuers in a low resolution grid, as the results in the simulations from Section 5.3 confirm.

The implementation of the sensing was limited by the available capabilities in the robot, as the camera only covers a small angle, and the acquisition of each image requires time. A different array of sensors, like a combination of cameras and ultrasonic sensors, would allow improving the speed of the sensing and processing, but this improvement can result quite expensive. The analysis of the environment can be adapted to work with beam sensing in fewer directions, to use the available camera in the robot, and reducing the time of computation.

The e-Puck platform comprises several disadvantages caused by the limited capabilities provided sensors, thus complicating the computation of the analysis of the environment. Also, following the identification of colours is a lengthy and inefficient process if computed every time for all the images from the sensing. Other alternatives are training a neural network to analyse the composition of the pixels in an image and determine the presence of elements accordingly. Also, the identification of elements in the grid did not consider the presence of walls in whiter colour as obstacles, which needs to be addressed as part of the sensing.

The overall motion of the robots is performed in a reasonable amount of time, as the low level control for the motors produces fast motions. However, a smoother motion would be more desirable, by joining the discrete planned path with a more continuous optimised one. Using the centroids of the cells as desired constraints for the motion, the optimisation of a continuous path could consider the dynamic constraints of the robot, in terms of steering and maximum speed.

In terms of the simulations, the environment was a limited sample of what it would be a realistic task. Nevertheless, a larger space with more complex obstacles and more pursuers with different behaviours is needed to assess the performance of the robot. A last form to improve the deliberation is the implementation of all the algorithms directly in the microcontroller, eliminating the communication with the computer to reach MATLAB.

8.5 Concluding remarks

This chapter demonstrated the implementation of some of the proposed designs in a robotic platform, the e-Puck. The main characteristics, advantages and disadvantages of the robot from the perspective of the application to the problem of navigation and escape from pursuers, presented in this thesis, were highlighted.

Subsequently, the proposed sensing process to identify the elements (obstacles, pursuers, free space and unknown cells) from images taken in an environment purposely built was presented. The coupling between the sensing, the proposed algorithms for the planning of a navigation path according to the objectives, and the low level control of the motion was explained. Visual examples of the resulting navigation, with one pursuer following the robot in the built environment, illustrated the functioning of the global control system.

Finally, the discussion highlights the main findings that confirm expected results from the simulations. The robot performs the motions according to the analysis and deliberation from the sensing through MATLAB, although the process of sensing to achieve a 360 degrees recognition of the surroundings is very slow. Possible improvements for the implementation are briefly mentioned, including the use of different sensors, implementing the algorithms directly in the microcontroller and training a neural network to perform the recognition of elements in the environment. Some of the proposed improvements to the platform to facilitate the sensing and processing can result quite expensive, but some software solutions in the form of changes to the environment analysis could help to avoid higher costs.

The next chapter presents the global conclusions of the thesis, highlighting the main contributions and the results from the proposals to solve the problem of navigation and escape from pursuers. Also, possible extensions of the project to future work are explained.

Chapter 9

Conclusions

This thesis presented different solutions to the problem of navigation and escape from pursuers, to avoid collisions with obstacles and prevent the capture for as long as possible. The solutions incorporated biologically inspired features in the form of refuges (hideaways) and proteanism (through subgoals for navigation). An analysis of the environment that incorporates the bio-inspired features was proposed, considering available sensed information, to compute local navigation goals. Two different approaches were proposed to compute paths or navigation strategies: path planning with potential functions and a modified steepest descent, and reinforcement learning with tabular Q-learning.

This chapter presents the final conclusions of the thesis, and possible future work to extend the project beyond its current frontiers. Throughout this thesis, the following topics were reviewed:

- Bio-robotics, with emphasis in the concepts of biologically inspired robotics and biomimetics, the degrees of inspiration incorporated in previous robotics projects, and areas of application through several examples. This part of the literature review allowed setting the context of the biological inspiration for the project.
- Mathematical ecology and behavioural biology, focusing on finding models of individual anti-predator motion behaviours that could be used directly along with control systems or other algorithms for autonomous navigation in robotics. Some features found in the review of anti-predator behaviours (like refuges and proteanism) and a couple of models, served as inspiration for the designs of solutions to the problem of navigation and escape from pursuers.
- Path planning methods, with emphasis in potential functions. The review allowed a comparison between different path planning methods, reaffirming the adequacy of potential functions for the design of solutions to the problem: potential functions do not require a complex geometrical analysis of the environment, they are fast and easy to implement and understand, and they can deal with environments of local incomplete information like the ones in the problem. From this review, different particular function implementations were chosen for the designs, trying to minimise the

problems of local minima previously reported in literature. Also, the search method of gradient descent was chosen considering the particularities of the environments of the problem.

- Pursuit-evasion games, focusing on proposed solutions to the evasion side, independently from the pursuit and for discrete, limited information environments. The use of reinforcement learning and the design of controllers for individual navigation were of particular interest.
- Reinforcement learning, considering an overview of the main algorithms, their design characteristics and features, and their use for autonomous navigation. Tabular Q-learning was chosen from the review, due to its advantages with respect to the particularities of the environment of the problem. The learning parameters were selected in a biologically inspired manner (particularly the reward function), considering their importance in the performance of the learning algorithm.

The designs have been inspired by the concept of *intelligent evasion*, a fusion of proteanism from the model of Furuichi (2002) and other studies on: co-evolution of navigation in prey and predator, evasion strategies as a response to pursuit in pursuit-evasion games (Pachter and Yavin, 1986; Amin et al., 1997; Murrieta-Cid et al., 2007; Karaman and Frazzoli, 2010; Shinar et al., 2010), path planning against pursuit (Masoud, 2002, 2003a,b) and stealth navigation. Proposing a predefined form to avoid pursuit and collisions provides a simplistic equivalence to the abstract concept of *innate self-preservation behaviours*, that could be commuted or discarded when the robot has developed its own mechanisms, behaviours or controls to guarantee its success when interacting with the environment.

The problem of navigation and escape was stated from two perspectives: *(a)* path planning, and *(b)* as part of a pursuit-evasion game, both when the environment is locally observed, discretely sensed and with incomplete information. Also, the pursuers are unknown but the robot might be able to use a worst-case scenario to analyse and predict their behaviour. From those perspectives, two main groups of designs to compute solutions to the problem, in terms of navigation plans or strategies, were proposed:

1. Path planning with potential functions.
2. Reinforcement learning with tabular Q-learning.

Different variations were proposed and compared from an evaluation: kinds of potential functions, frequency to update the sensing of the environment, the computation of proteanism (guided, general, long or short reaching), kinds of reward function, and parameters for the learning. A statistical analysis of the success of computed paths in navigation and escape, for a simulated setting, allowed comparing the variations previously mentioned, and in general the performance of the proposals. The following subsections present the final conclusions for the main components of the designs:

- The analysis of the environment (Section 9.1).
- The solutions using path planning with potential functions and *a priori* proteanism (Section 9.2).

- The solutions using Q-learning and shaped reward functions (Section 9.3).

9.1 Biologically inspired analysis of the environment

The analysis of the environment performs the following actions:

- Produces a probability map for the sensed elements in the environment, identifying the components into pursuers, obstacles, free space or unknown space, in a discrete form.
- Extends probable obstacles and pursuers, according to the criteria of the designer.
- Increases the resolution of the grid artificially, towards a more continuous representation and computation of the navigation
- Computes a local goal for navigation in the locally sensed environment, from functions of distances to pursuers, to move away, and biologically inspired hideaways.
- Implements biologically inspired *a posteriori* proteanism, by computing randomly selected subgoals towards the local goal.
- Converts the occupancy grid, the local navigation goal and or the subgoals into a potential functions based representation.

The proposed analysis of the environment has the following advantages with respect to the problem of autonomous navigation and escape, considering locally sensed information only:

- Allows the integration of sample-based information from the sensing, into an approximate cell decomposition or occupancy grid representation. This reduces the need of large amounts of data to compute a more precise geometrical representation of the obstacles, thus being able to deal with information from basic sensors like infrared or ultrasonic, instead of high resolution cameras and complex image processing.
- Also, the proposed probabilities for each cell allow integrating the inaccuracy of the information provided by the sensing.
- The use of a grid representation, like the proposed, can be adapted to different path planning methods, depending on the resolution of the grid and through a geometrical processing. For example, information coming from a fine resolution grid can be transformed into connections of lines for a visibility based analysis, and graph theory can be used to search for optimal solutions.
- The computation of local navigation goals as functions of distances to pursuers and possible hideaways provides general guidance for the navigation of the robot in the local environment. The local goals are convenient to decompose a longer global navigation task into dealing with locally available information at a time.
- A natural reaction to pursuit is navigating as fast as possible in the same direction as the pursuit is taking place. This would guarantee an escape if the evader is faster

than the pursuer. Otherwise, another alternative to pursuit is changing the angle of the linear motion abruptly to make the pursuer steer and waste time by correcting its motion again towards the evader (Isaacs, 1965). The problem increases its degree of complexity when multiple pursuers are around, coming from different parts of the environment. The proposed analysis of the environment offers the computation of the best location to move in terms of trying to avoid the pursuers by moving to a place far away from all of them at the same time, and at the same time trying to seek temporal refuge behind obstacles. This combination of concepts mixes stealth navigation with evasion ideas and anti-predator behaviours.

- The use of subgoals to lead towards a local goal accomplishes two main functions in one. Subgoals help to prevent local minima problems recurrently reported in literature review for potential functions (Bell, 2005), besides being a convenient implementation for the *a priori* proteanism.
- The representation of the environment can be used in combination with any robotic platform and sensing capabilities, by modifying the shape of the grid and the resolution of the cells accordingly.

The biologically inspired analysis of the environment is advantageous in terms of the intrinsic generality of its design. The analysis can be adapted for its application to different robotic platforms, to deal with locally sensed environments when no previous information is available. Also, the analysis can be extended to other path planning techniques, and different kinds of models for its use in control theory. The implementation of the analysis is easy to understand, and computationally fast.

The following aspects of the analysis of the environment can be improved:

- The analysis of the environment does not contemplate a mechanism to achieve the identification of threats from obstacles or other not dangerous elements. This step can be improved by proposing a recognition algorithm, that classifies the elements in the environment into static or dynamic obstacles and possible threats.
- Other kind of environment representations can be used from the acquisition of data, without prior conversion into a grid, like an exact cell decomposition according to the observed geometries, or Voronoi diagrams (Choset et al., 2005). Different environment representations naturally lead to the use of alternative path planning algorithms.
- If robotic navigation behaviours are to be implemented to approximate specific or general animal anti-predator behaviours, more mathematical and conceptual models are needed, extending the work by Furuichi (2002) and Broom and Ruxton (2005).

9.2 Path planning solutions

The proposed path planning solutions considered the biologically inspired analysis of the environment mentioned previously, and also the following:

- The use of potential functions to compute a value for each cell in the analysis of the environment, in terms of the presence of obstacles, pursuers, the local goal and *a priori* proteanism (subgoals). A search method deliberates the path by following the optimal values. Potential functions in a discrete representation do not require a complex analysis of the geometries in the environment.
- A modification to the steepest descent search method, to consider the shortest distances as a second criterion when the values of the available choices are equal. Evaluating the distances is useful for different kinds of local minima problems commonly encountered in potential functions for path planning.
- Three main kinds of paths were proposed to result from the combination of the analysis of the environment and search method: guided protean, general protean and steepest descent only. Each one of them depending on the optional use of subgoals, replacing temporarily the local navigation goal, to prevent a strictly linear navigation towards the local goal.

The performance of the proposed paths was evaluated through a simulation process, with randomly generated environments and pursuers modelled as followers. Variations in the number of pursuers, their velocities, updates of the sensing, grid resolutions and radius of computation of subgoals, were analysed to deliberate the best combinations according to the simulated characteristics of the environment. The relative and absolute success of the paths and other statistics on the navigation steps before capture, served as the basis for the evaluations.

The results indicate that the ideal combination of parameters, particularly the updates of the sensing and computation of subgoals to increase the success avoiding capture, depends on the number of pursuers and their velocity. For few pursuers, a more constant update of the sensing is needed to detect and respond to the threats. For many pursuers (in a cluttered environment), protean paths are a good resource to avoid capture, although it is almost impossible to escape. A balance in the frequency of the sensing updates and the use of a computed path is needed, to use the latter in a more efficient manner.

Steepest descent paths, in combination with modified FIRAS and quadratic goal potential functions, performed better than any other combination, for all numbers of pursuers, their different velocities, and the updates in the environment. The visual examples showed that some resulting paths are more linear, and others more protean, according to the presence of pursuers, and the randomness of the subgoals. The proposed computed paths work better in higher resolution grids, according to the analysis. The computed paths are not entirely successful, as the simulations demonstrated. This situation makes evident the dependence of the avoidance of capture not only on the number of pursuers, or their speed, but also the initial locations of the pursuers, and even the location of the obstacles.

The proposed paths, based on potential functions, have the following advantageous features towards the solution to the problem of navigation and escape from pursuers:

- Potential functions are conceptually and computationally easy to implement, providing with a fast deliberation of the paths according to the analysis of the environment.

- The use of subgoals when computing a path serves two main purposes: changing the direction of the motion of the robot to disorient pursuers and preventing the presence of local minima problems in the potential functions.
- The discrete resulting path can be interpreted as moving towards a region. Continuous paths that connect the regions or general points in the grid can be computed considering the kinematic and physical characteristics of a robot, by coupling a low level smoothing control system. On the other hand, increasing the resolution of the grid artificially can be used to compute more continuous paths, as needed according to the robotic platforms.

Some proposed improvements for the path planning using potential functions are:

- A structure that chooses between steepest descent only or subgoal based paths, frequent or sparse sensing updates and the effect of the proteanism (due to the radius for computation of the subgoals). This structure can be implemented as a control system to deal with different situations in the environment. The ideal parameters would be chosen according to the number of pursuers, their locations, and their probable velocities.
- Another aspect that can be improved is the smoothness of the final path, which could be added as part of the path planning instead of leaving it to a low level control system. *a posteriori* proteanism, and kinematic considerations could be added for a smoother and improved path.
- More information about the pursuers is needed, coming from the environment analysis to identify an approximation of their velocities that would determine the characteristics of the computed path.
- Several options can be used to substitute the potential functions presented in Chapter 4. Strong candidates are vector potential fields incorporating more complex behaviours like rotations around obstacles. Other related path planning methods from the ones mentioned in Chapter 3 can be used to evaluate their performance compared to potential functions in the particular task of navigation and escape from pursuit.
- The information from the environment analysis and the resulting paths can be stored in memory, to create a library of previous situations and reuse improved solutions. This recycling would lead to a more efficient deliberation process.
- Different simulations to evaluate the path planning proposals are needed, including more realistic and extended environments.

9.3 Reinforcement learning solutions

The proposed reinforcement learning solutions also considered the biologically inspired analysis of the environment mentioned previously, along with the following:

- A tabular Q-learning algorithm to compute a navigation strategy, from simulated interactions with the environment. Q-learning calculates the action-value function for all the combinations of visited action-state pairs, through predicting the following state transitions after choosing an action in a determinate state and receiving a reward.
- The selection of the initial learning parameters for the Q-learning algorithm (the learning rate, the discount rate, the probability for exploration and the number of iterations or episodes) are selected according to a measure of the danger in the environment, in function of the distances from the robot to all the identified pursuers in the local environment. The selection of dynamic parameters allows the computation of strategies favouring exploration or exploitation, with the consequent reduction in the total deliberation time, as required by the presence of threats in the environment.
- Two different reward functions were proposed, based on the computation of the local navigation goals in the biologically inspired analysis of the environment. The first reward function incorporates the interaction with the elements in the environment, as the consequence of the transition state, in the form of an if-else structure (shown in a tabular form). The second reward function has been based on potential functions, and incorporates functions of the distance to the local goal and the pursuers and obstacles in the environment.

The ranges of operation for the parameters have been determined from experimentation, by computing the average RMSE and the average deliberation time when varying some parameters at a time. The simulations also revealed the importance of the balance between exploration and exploitation for the task of computing navigating strategies that lead towards a local goal.

As previously, the performance of the proposed strategies was evaluated through a simulation process, with randomly generated environments and pursuers modelled as followers. Variations in the number of pursuers, their velocities and updates of the sensing, were analysed to deliberate the best combinations according to the simulated characteristics of the environment. The relative success of the paths, for the different reward functions, and statistics on the navigation steps before capture, served as the basis for the evaluations.

Compared to path planning, the navigation strategies use more time in the computations. Furthermore, the success of the paths computed with reinforcement learning is lower than the path planning paths, in terms of the number of navigation steps before capture.

The reinforcement learning proposed approach has the following advantages, with respect to the problem of navigation and escape from pursuers:

- Intrinsic to reinforcement learning, one of the advantages is that no mathematical model of the environment or elements is needed to compute the optimal strategies, but information about the state transitions and the corresponding reward. This situation is ideal to deal with previously unknown environments, from which only some information is available through local sensing.

- Tabular Q-learning can be extended to deal with more data, from higher resolution grids or a more complex sensing source, by means of function approximation techniques.
- Reinforcement learning algorithms, like the tabular Q-learning, are easy to understand and implement computationally.

The reinforcement learning approach can be improved in the following ways:

- A more biologically based approach to avoid pursuit, similar to Furuichi (2002), can be considered for the reward functions, to substitute the need for a local navigation goal and promote the use of proteanism according to the conditions in the environment. Consequently, the approach could change from the computation of navigation strategies towards a goal, to a more reactive form to deal with pursuit.
- Other reinforcement learning algorithms, like Sarsa (λ) or an actor-critic architecture, can be used to substitute Q-learning, based on a similar dynamic selection of parameters and the same reward functions. The evaluation of different reinforcement learning approaches in the solution to the same problem of navigation and escape would allow producing the best combinations according to the requirements of the task.
- Additional improvements include smoothing the final strategy, and acquiring more information about the pursuers, as mentioned before for the path planning approach.

9.4 Implementation in robots

The demonstration of the path planning approach in a robotic platform, the e-Puck, required the following additions:

- A program for the micro-controller in the robot, to perform the sensing of the environment through taking 8 photos with the camera included as part of the hardware of the robot. The photos are transferred to MATLAB for the extraction of elements and distances.
- A program to process the images from the sensing, identifying pursuers, obstacles, free space and their relative distances (with respect to the robot). An analysis of colours and brightness is used for the identification of the elements, and the output is an occupancy grid to be used in the environment analysis described in Chapter 4.
- A low level control that receives the coordinates (x, y) of the following cell in the path (its centroid), and feeds the robot the corresponding output signal in terms of the speed of the two wheels, and the time to sustain that speed. The robot executes the motion in two steps: correcting the angle first and then a displacement in the direction of that angle, for the approximate length of a cell.

The simulations with the robot show that the elements are identified correctly, and the occupancy grid is computed accordingly. Nevertheless, the capabilities of the camera are not fully used, as no information about the geometry and other features of the pursuers are considered in the analysis. Also, the procedure to acquire information about the surroundings at once is limited due to the small angle of range in the camera. A different array of sensors would provide with an equivalent amount of information in less time, and without the need to stop and move the entire robot to perform the sensing.

The communication between the robot and MATLAB, via Bluetooth, transmits the sensed images from the robot and receives a path in return relatively fast. The resulting motion towards the cells specified in the path is also fast, using the full capabilities of the robot in speed terms, and is accurate for a maximum of 8 navigation steps, after which a correction in the location of the robot is needed. However, physical constraints can be considered to smooth the path from the path planning processing, towards a continuous motion that changes the angle whilst moving towards the target location.

9.5 Main achievements

The main achievements of the work presented in this thesis are the following:

- The development of a biologically inspired analysis of the environment, where information is processed to determine a local navigation goal, *a priori* proteanism and a representation of the environment.
- A proposal to consider not only a location away from all pursuers simultaneously as a local goal for navigation to solve the problem of escape, but considering biologically inspired temporal refuges behind obstacles to try to improve the chances of avoiding capture.
- A proposal to implement *a priori* and *a posteriori* proteanism as part of the autonomous navigation of a robot, and to solve the problem of escape from pursuers.
- A different use of subgoals in navigation, not only to solve the problem of local minima in potential functions for path planning, but to compute *a priori* proteanism.
- The use of two different approaches to propose solutions to the problem of navigation and escape from pursuers:
  - A path planning perspective, based on potential functions.
  - A reinforcement learning perspective, based on tabular Q-learning.
- The incorporation of pursuers into different kinds of potential functions, as well as the subgoals for guided *a priori* proteanism.
- A modification of traditional steepest descent search method, to compute the path planning from the potential functions environment representation, by incorporating a second criterion based on the distances to the local goal or currently active subgoal, to facilitate the computations.

- The selection of learning parameters for a reinforcement learning algorithm based on the closeness of threats, to deliberate a navigation strategy as soon as possible.
- The shaping of reward functions to facilitate the computation of navigation strategies, from the concept of local navigation goals to navigate far away from pursuers, and biologically inspired refuges.
- The design of simulation tools in MATLAB for the sensing acquisition, the deliberation and the pursuers, to evaluate the performance of the proposals from a visual perspective.
- The proposal of an evaluation system of paths and navigation strategies, from the simulated sensing and interaction with pursuers in randomly generated environments. The evaluation system is based on a statistical analysis of the success of the paths avoiding capture in a threshold of navigation steps.
- The generation of control systems for navigation and escape from pursuers that can be used for different robotic platforms.
- The demonstration of the functioning and implementation of the proposed path planning control system in a robotic platform, the e-Puck. A sensing processing and a low level control were proposed to be coupled with the biologically inspired environment analysis and the path planning based on potential functions.

9.6 Future work

The following subsections detail three main areas of possible future work to extend the proposals of this thesis to a specific robotic platform, a more complete simulator that allows evaluating the proposals, and towards multi-robot systems.

9.6.1 A solution to navigation and escape specific to a physical robot model

The work presented in this thesis can be improved by considering the characteristics of a specific robotic platform to be used in another real life task, with a design based on the same elements in the control system described in Chapter 8. The proposed improvements are:

- Taking better advantage of the capabilities of the sensing hardware, in terms of the extraction of information. When more information is available, mathematical models can be computed and used in the deliberation process to improve the responses for the navigation and escape from pursuers. The representation of the environment needs to reflect as much available information as possible, but considering the computational cost and time limitations.

- Considering relevant internal variables of the robot as part of the sensing, like the state of the battery, when they affect the performance of the navigation.

- Using self-localisation and mapping (SLAM) as part of the sensing, to accumulate more knowledge about the environment in memory. The accumulation of knowledge would lead to navigation strategies that consider more options in terms of future possible locations for the robot, but also has a computational cost by increasing the configuration space and requiring more memory.

- Applying smoothing techniques considering the physical characteristics and kinematic limitations specific to the robot hardware, to improve the discrete paths.

- Improving the architecture of the control system according to the requirements of the task, as the combination of collision avoidance and escape from pursuers with exploration, area sweeping or transportation could require a different set of actions. The proposed navigation and escape solutions can form part of a modular or hierarchical structure that switches between modes of operation according to the current objectives. Furthermore, the proposals of this thesis could be implemented as part of the pre-defined responses when the environment is previously unknown, for a control structure that learns behaviours or actions from the interactions with the environment.

- Implementing proteanism in a different form, combined with stealth navigation to complement the bio-inspired notion of refuges, according to the chosen environment representation.

- Incorporating mechanisms to model the pursuers to predict their behaviour, when no previous information about them is available. Mathematical models of the pursuers can be built along with the sensing of the environment, and modified online according to the observations from the interactions. Another proposal is a global model that concentrates the characteristics of different kinds of pursuers from the literature (like the ones in differential pursuit-evasion games) can be designed and embedded *a priori*.

A global criterion to guide the design of the different stages in the control system (sensing, deliberating and moving or acting), is the specific needs and challenges of the environment. For example, if the robot is deployed in an area where the pursuers and obstacles are of certain type, like surveillance in a floor of a building, the design of the control system can contemplate the use of a predefined solution. If the robot was to be deployed for exploration in another planet, a more robust and complex system that adapts to the new challenges in the environment would be a better solution, to let it learn from its interactions. A comparative study of all the available options for the structure components in a control system for more autonomous navigation and escape from pursuers is needed, to find the best alternatives and improve them as much as possible in combination with other real life tasks like surveillance, search and rescue, cleaning, or recognition. A start of such a comparative study can be found in Amin et al. (1997).

9.6.2 Improving the methodology to evaluate solutions to the problem

Ideally, a more complete simulator would lead to a better evaluation of any approach to the solution of the problem of autonomous navigation and escape from pursuers. The characteristics of an ideal simulator would be:

- The use of structured environments and terrains, where the obstacles have complex geometries, and its extension is not necessarily limited by walls.
- Allowing a better visual representation of the motion of the robot with respect to its environment including the steering angles, velocities (linear and angular) and accelerations.
- A set of predefined pursuer behaviours, from common pursuit strategies from the literature, to be able to combine them in the same environment and assess the performance of the navigation and resulting behaviours caused by the pursuit.
- Incorporating many other modular path planning, machine learning and control systems components, programmed previously. This would allow assembling different combinations of control structures, to observe the consequent navigation behaviours due to the interaction with the artificially created environment.
- Having a visual interface where paths or navigation strategies computed with different methods are compared through relevant statistics, for the same simulated environment. The user would be able to input the characteristics of the statistical analysis, like number of samples, measures to assess the differences between the paths and measures of the success in both avoiding collisions and avoiding capture by pursuers.

Some of the proposals to improve the simulating environment have been inspired in other available robotics simulators, and the modularity of toolboxes in software like MATLAB or LabVIEW (Ponce-Cruz and Ramírez-Figueroa, 2009).

9.6.3 Extension to multi-robot systems

The proposed solutions to the problem of navigation and escape from pursuers can be directly extended to the multi-robot systems domain, by adding multi-agent systems concepts (Weiss, 1999; Wooldridge, 2002). Some suggestions are listed next:

- Communication between the robots that conform the multi-robot system, to share information about the environment or about themselves (like their relative location, or the result of their own decision processes) and for coordination purposes. Some multi-robot control systems do not need communications, as each robot obeys its own control system and decides according to its own observations. Nevertheless, communications offer an easier alternative than modelling and predicting the behaviour of other robots that form part of the multi-robot system.
- Coordination and negotiation mechanisms to decide on individual local goals of navigation, the grouping of the robots into coalitions and the division of the tasks. A

coordination and communication mechanism is needed if the robots decide about their own navigation considering the decisions of the other robots, in a cooperative form. Popular coordination and negotiation mechanisms include bidding and voting (Partalas et al., 2007). The formation of coalitions simplify the task of navigation, by creating hierarchies for the planning. The task of escaping from multiple pursuers can be divided in simpler tasks, where a pursuer is assigned to a group of escaping robots, trying to maximise the chances of avoiding capture by being in a group.

- Swarm concepts like formations and clustering can be incorporated to facilitate the coordination of the navigation, as some robots could follow a leader that decides the main escape strategy. The use of leader-follower hierarchies and the separation of the whole population of robots in the system into groups, allows the incorporation of biologically inspired group behaviours like mobbing or group proteanism.

Considering multi-robot systems, or the presence of allies in the environment, increases the complexity of the collision avoidance. The allies are dynamic entities, with their own intelligence and motion capabilities. Thus, the multi-robot system needs constant communication between the robots to know of the actions of others, or keeping a model of the behaviour of the other robots and considering predictions about their navigation in the deliberation process. Otherwise, avoiding collisions with the allies without knowing about their actions leads to more pressure in the sensing stage.

Appendix A

Assessment of similitude between two curves

Discrete Fréchet distance

The discrete Fréchet distance is a measure of similarity between two curves (the paths) (Eiter and Mannila, 1994). Consider two polygonal curves, Path1 and Path2, with q_1 and q_2 relevant points (vertices) each. The coupling of the curves is defined as a sequence $(\text{PathA}_1, \text{PathB}_1), (\text{PathA}_2, \text{PathB}_2), ..., (\text{PathA}_{q_1}, \text{PathB}_{q_2})$. The length of the coupling is defined as

$$||L|| = \max_{i=1,\ldots,q_l} d(\text{PathA}_i, \text{PathB}_i), \tag{A.1}$$

where L is the longest link between the two curves, $d(\text{PathA}_i, \text{PathB}_i)$ is the distance between the points in the polygons and q_l is the maximum number of points in any of both curves (q_1 or q_2). Thus, the Fréchet distance between the two curves is

$$\delta_{DF}(\text{PathA}, \text{PathB}) = \min ||L|| \tag{A.2}$$

or the minimum of all possible links between the curves. The algorithm used to compute the discrete Fréchet distance, from Eiter and Mannila (1994), is shown next.

Root-mean-square error

The root-mean-square error is a measurement of the accumulated difference between the points of the subgoal based protean path and the steepest descent only path,

$$\text{RMSE} = \sqrt{\frac{\sum_{m=1}^{q_l} \left(d(\text{PathA}_m, \text{PathB}_m)\right)^2}{q_l}} \tag{A.3}$$

Algorithm 5 Discrete Fréchet distance algorithm

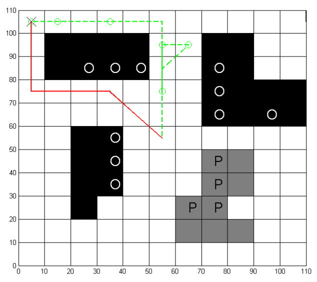
```
1: Initialise $G(j,k) = -1 \forall j,k$ as a matrix of $q_1 \times q_2$ dimensions, $j = 1,...,q_1$ points in PathA, $k = 1,...,q_2$ points in PathB
2: $G(1,1) = d(\text{PathA}_1, \text{PathB}_1)$
3: for $j = 1 : q_1$ do
4:     $G(j,1) = \max\{G(j-1,1), d(\text{PathA}_j, \text{PathB}_1)\}$
5: end for
6: for $k = 1 : q_2$ do
7:     $G(1,k) = \max\{G(1,k-1), d(\text{PathA}_1, \text{PathB}_k)\}$
8: end for
9: for $j = 2 : q_1$ and $k = 2 : q_2$ do
10:    $G(j,k) = \max\{\min\{G(j-1,k), G(j-1,k-1), G(j,k-1)\}, d(\text{PathA}_j, \text{PathB}_k\}$
11: end for
12: Fréchet distance = $G(q_1, q_2)$
```

where q_l is the largest number of points from any of the two paths, q_1 for PathA, and q_2 for PathB accordingly.

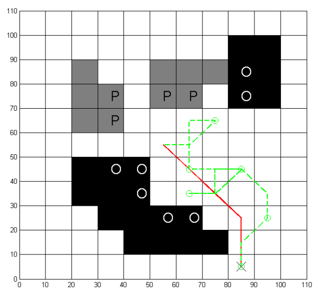
Appendix B

Additional visual examples for a low resolution grid

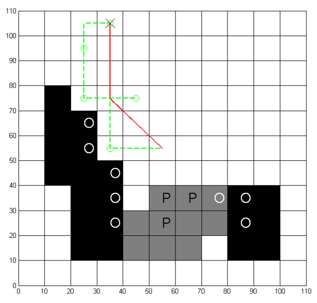
This appendix presents additional examples of computed paths, for a low resolution grid, in environments with two pursuers and different potential functions. The examples complement the simulations and results from Sections 5.2 and 5.3.



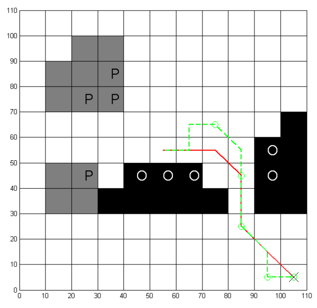
(a) Successful paths



(b) Successful paths

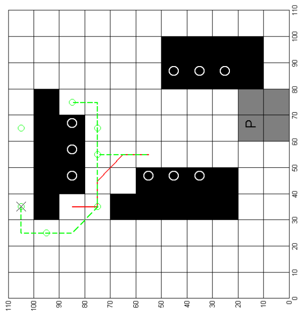


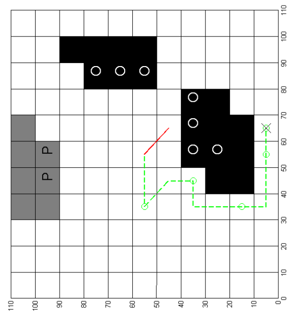
(c) Successful paths

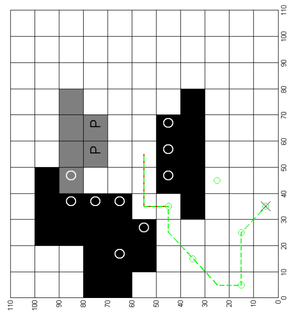


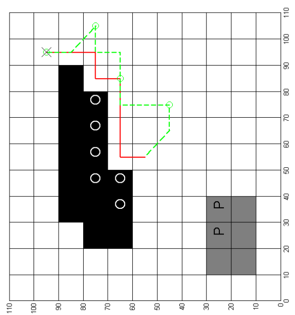
(d) Successful paths

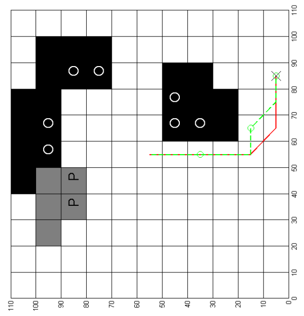
Figure B.1: Computed paths using steepest descent (solid red) and updated subgoals and steepest descent (dashed green), with subgoals as circles and modified FIRAS and quadratic potential functions; sensed obstacles in black with an O, sensed pursuers in grey with a P, extended probable obstacles in black, extended probable pursuers in grey, and the *LG* as X.

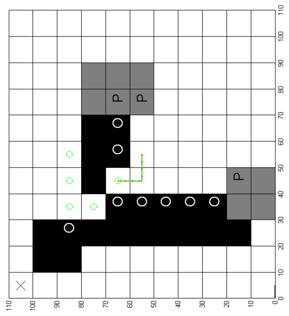

(e) Local minima in steepest descent path

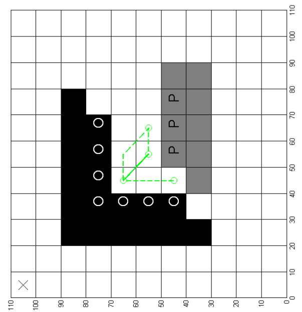

(f) Local minima in steepest descent path

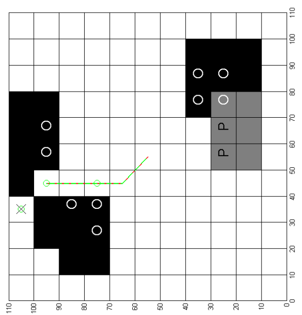

(g) Identical paths


(h) Successful paths

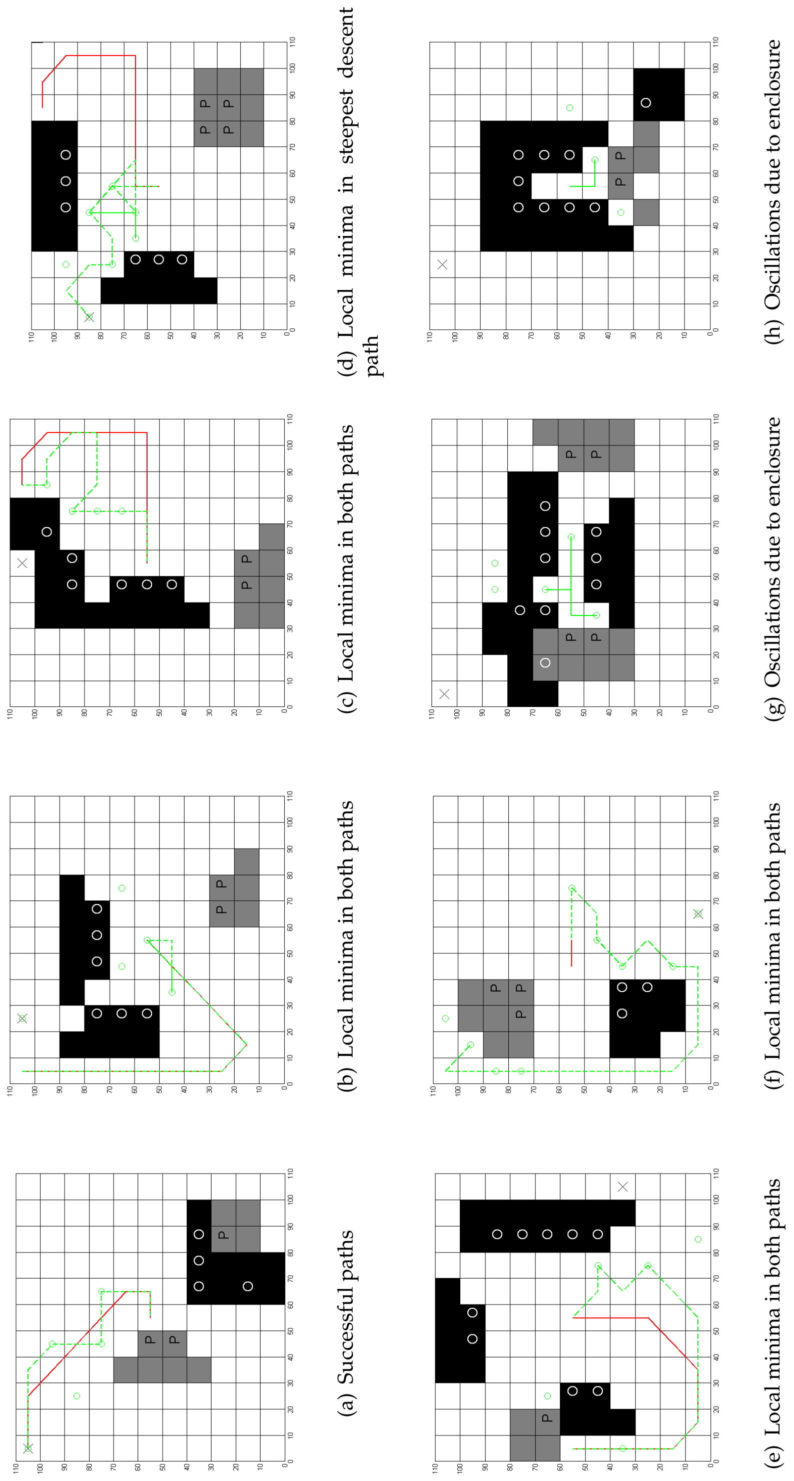

(i) Successful paths


(j) Local minima in both paths


(k) Oscillations due to enclosure


(l) Local minima in both paths

Continuation of Figure B.1



(a) Successful paths

(b) Local minima in both paths

(c) Local minima in both paths

(d) Local minima in steepest descent path

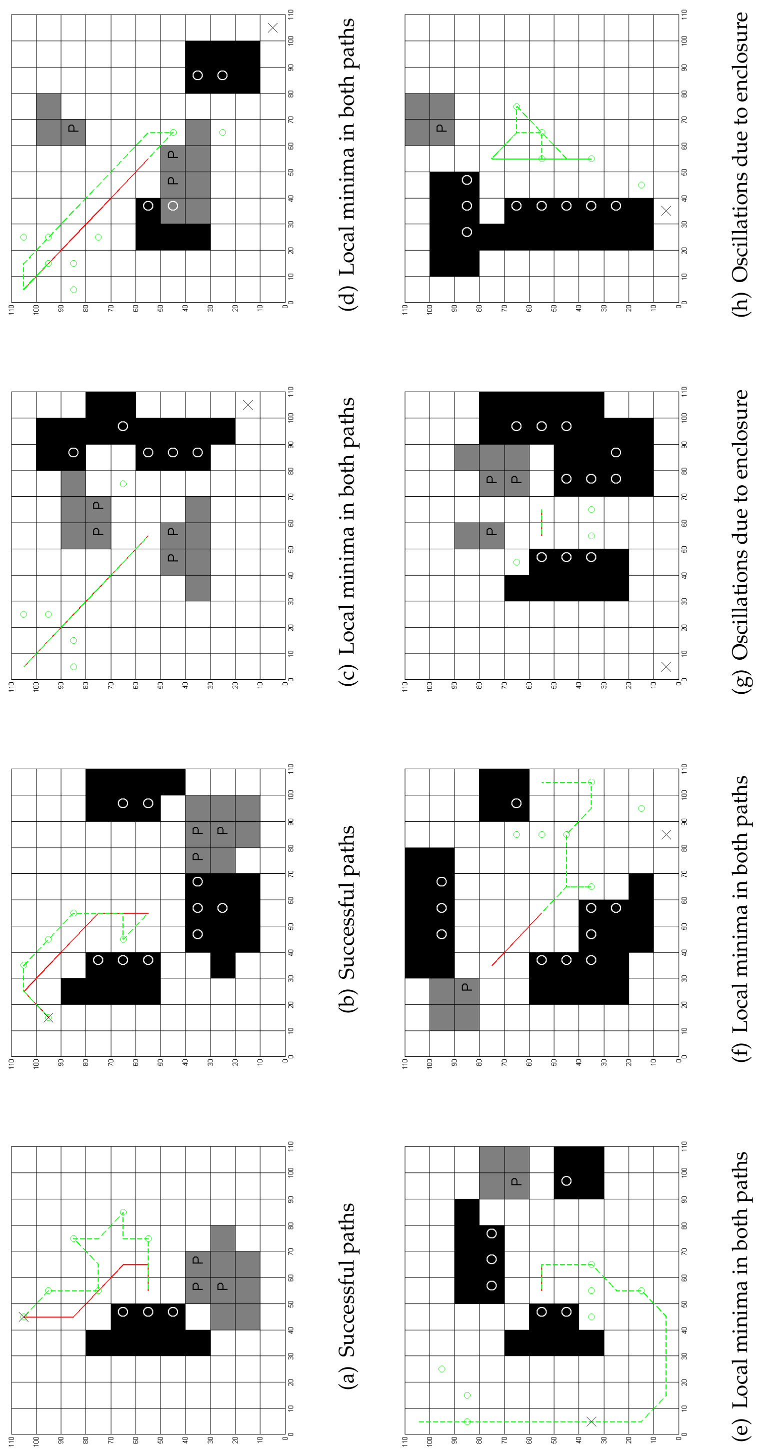
(e) Local minima in both paths

(f) Local minima in both paths

(g) Oscillations due to enclosure

(h) Oscillations due to enclosure

Figure B.2: Computed paths using steepest descent (solid red) and updated subgoals and steepest descent (dashed green), with subgoals as circles, and a BVP with Dirichlet conditions and Jacobi iterative method; sensed obstacles in black with an O, sensed pursuers in grey with a P, extended probable obstacles in black, extended probable pursuers in grey, and the *LG* as X.


(a) Successful paths

(b) Successful paths

(c) Local minima in both paths

(d) Local minima in both paths

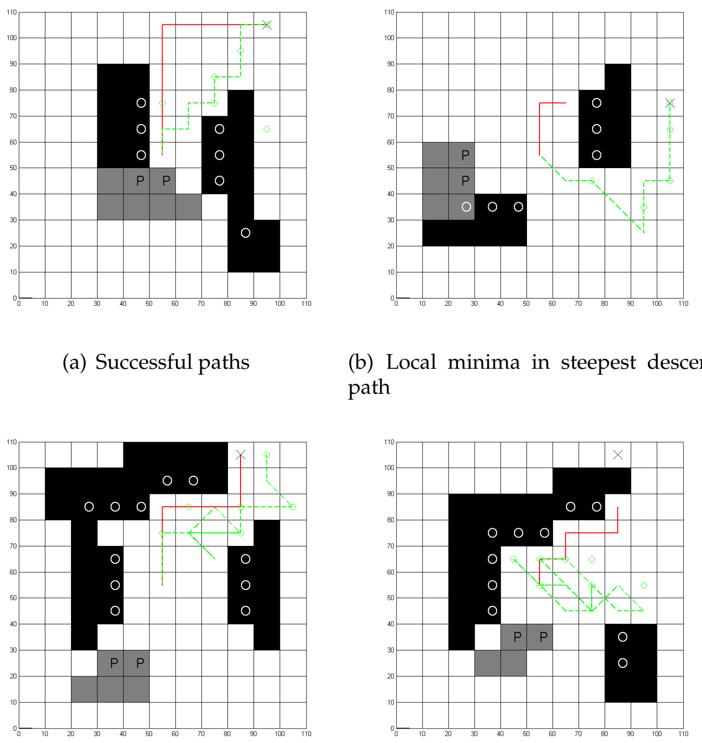
(e) Local minima in both paths

(f) Local minima in both paths

(g) Oscillations due to enclosure

(h) Oscillations due to enclosure

Figure B.3: Computed paths using steepest descent (solid red) and updated subgoals and steepest descent (dashed green), with subgoals as circles, and a BVP with Dirichlet conditions and SOR iterative method; sensed obstacles in black with an O, sensed pursuers in grey with a P, extended probable obstacles in black, extended probable pursuers in grey, and the *LG* as X.



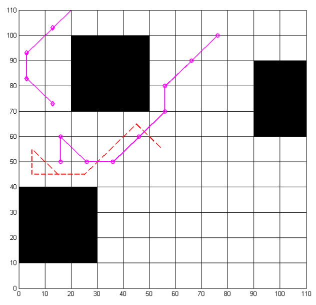
(a) Successful paths

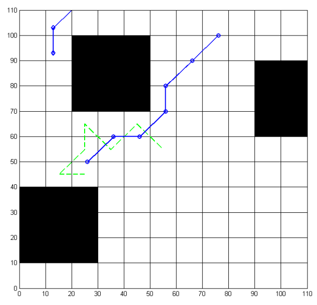
(b) Local minima in steepest descent path

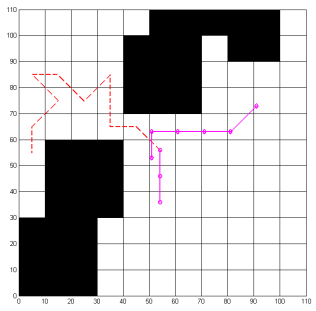
(c) Local minima in protean path

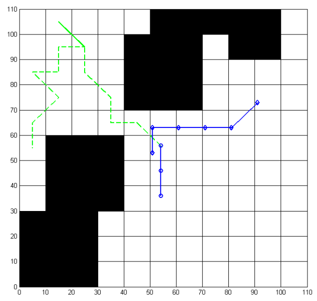
(d) Oscillations due to enclosure

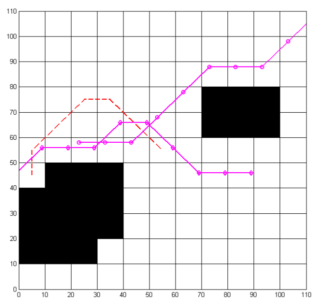
Figure B.4: Computed paths using steepest descent (solid red) and updated subgoals and steepest descent (dashed green), with subgoals as circles, wave-front potential functions; sensed obstacles in black with an O, sensed pursuers in grey with a P, extended probable obstacles in black, extended probable pursuers in grey, and the *LG* as X.


(a) Steepest descent path


(b) Subgoal based path


(c) Steepest descent path


(d) Subgoal based path


(e) Steepest descent path

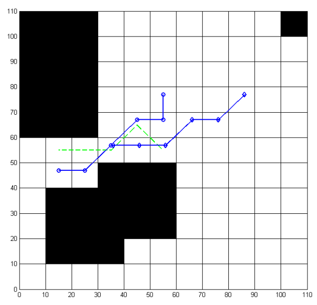
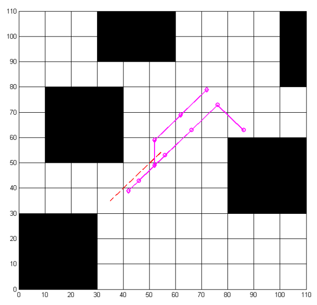
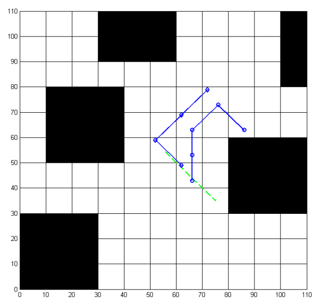

(f) Subgoal based path

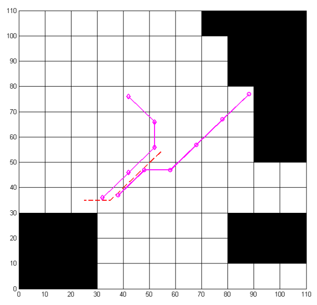
Figure B.5: Concentrated paths simulating the navigation of the robot and the pursuers with a velocity of $v_p = v_r$, and updating the sensing after every navigation motion. Steepest descent paths in red and corresponding pursuers in magenta; subgoal based paths in green and corresponding pursuers in blue; obstacles in black.



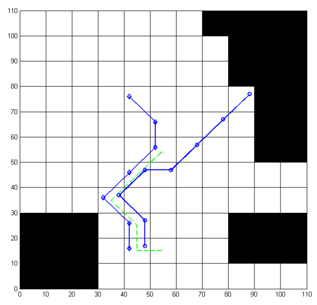
(a) Steepest descent path



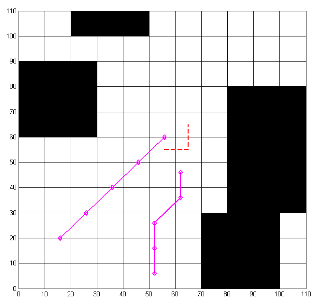
(b) Subgoal based path



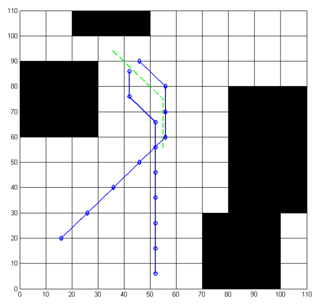
(c) Steepest descent path



(d) Subgoal based path

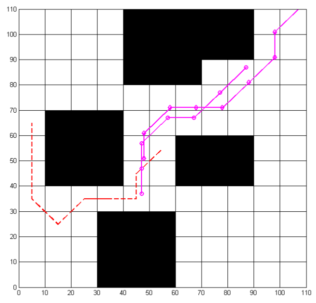


(e) Steepest descent path

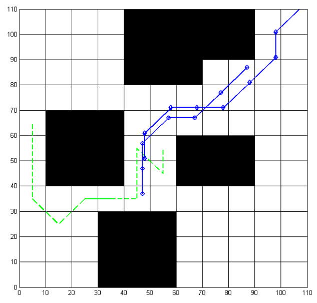


(f) Subgoal based path

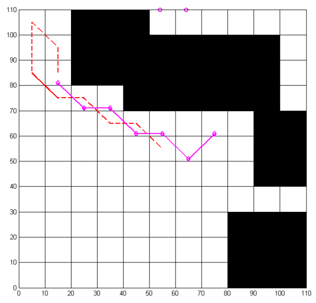
Figure B.6: Concentrated paths simulating the navigation of the robot and the pursuers with a velocity of $v_p = 2v_r$, and updating the sensing after every navigation motion. Steepest descent paths in red and corresponding pursuers in magenta; subgoal based paths in green and corresponding pursuers in blue; obstacles in black.



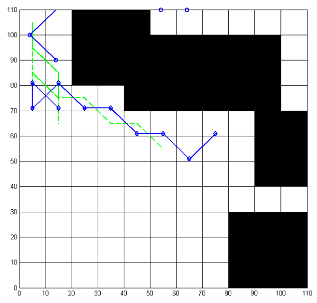
(a) Steepest descent path



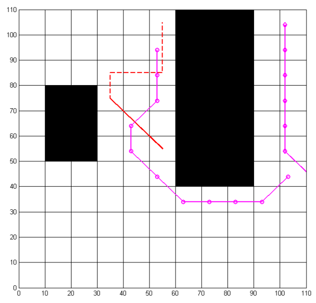
(b) Subgoal based path



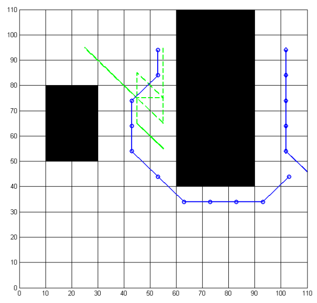
(c) Steepest descent path



(d) Subgoal based path



(e) Steepest descent path



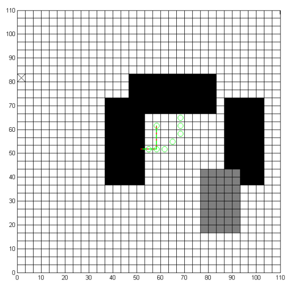
(f) Subgoal based path

Figure B.7: Concentrated paths simulating the navigation of the robot and the pursuers with a velocity of $v_p = \frac{1}{2}v_r$, and updating the sensing after every navigation motion. Steepest descent paths in red and corresponding pursuers in magenta; subgoal based paths in green and corresponding pursuers in blue; obstacles in black.

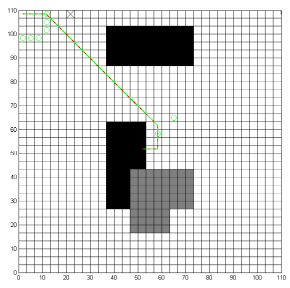
Appendix C

Additional visual examples for a high resolution grid

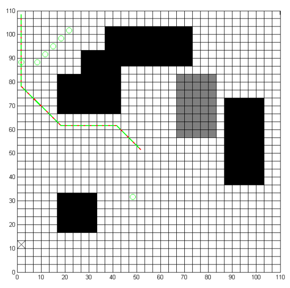
This appendix presents additional examples of computed paths, for a low resolution grid, in environments with two pursuers and different potential functions. The examples complement the simulations and results from Sections 5.4, 5.5 and 5.6.



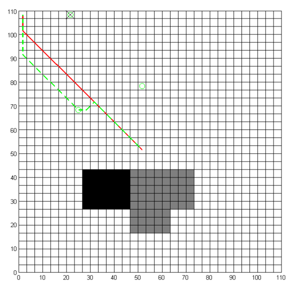
(a) Oscillations due to enclosure, $r_{sg} = 3$



(b) Local minima problems, $r_{sg} = 3$

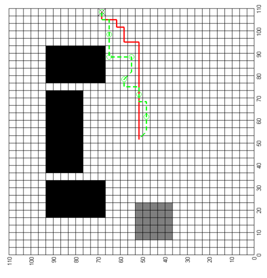


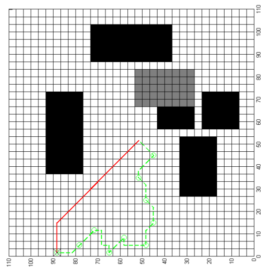
(c) Local minima problems, $r_{sg} = 6$

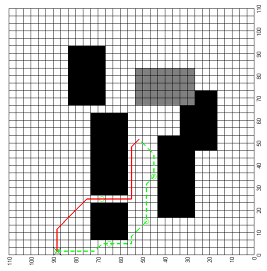


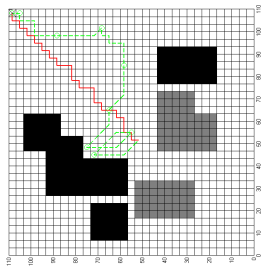
(d) Local minima problems, $r_{sg} = 9$

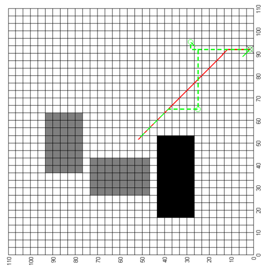
Figure C.1: Computed paths using steepest descent (solid red), and updated subgoals and steepest descent (dashed green), with subgoals as circles, and a BVP for Laplace equation with Dirichlet conditions and Jacobi iterative method; sensed obstacles in black, sensed pursuers in grey, and the LG as X.

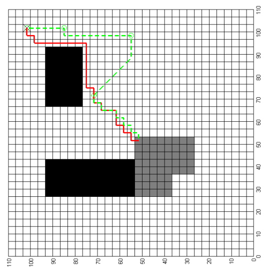

(a) Successful paths, $r_{sg} = 3$

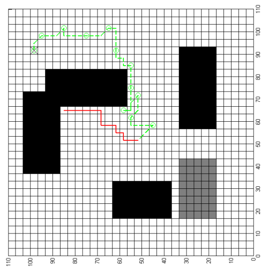

(b) Successful paths, $r_{sg} = 3$


(c) Successful paths, $r_{sg} = 6$


(d) Successful paths, $r_{sg} = 6$


(e) Successful paths, $r_{sg} = 9$


(f) Successful paths, $r_{sg} = 9$


(g) Local minima in steepest descent path, $r_{sg} = 3$

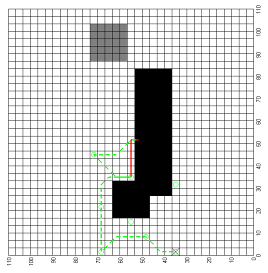
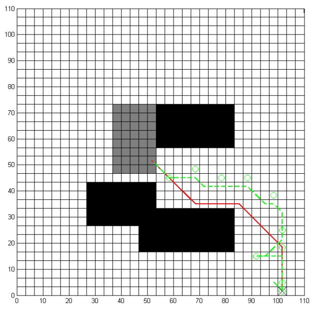
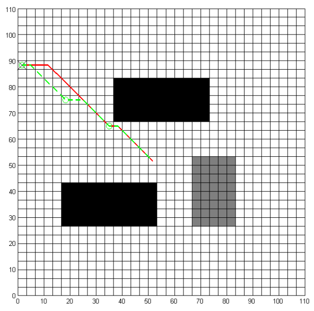

(h) Local minima in steepest descent path, $r_{sg} = 6$

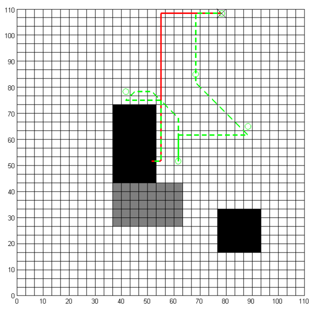
Figure C.2: Computed paths using steepest descent (solid red) and updated subgoals and steepest descent (dashed green), with subgoals as circles and modified FIRAS and quadratic potential functions for a highr resolution grid; sensed obstacles in black, sensed pursuers in grey, and the LG as X.



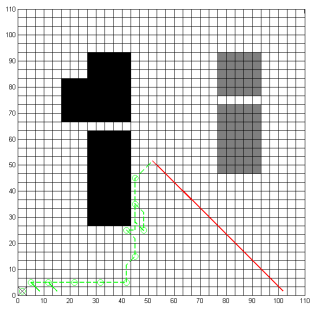
(a) Successful paths, $r_{sg} = 3$



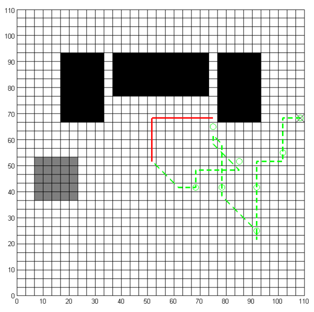
(b) Successful paths, $r_{sg} = 6$



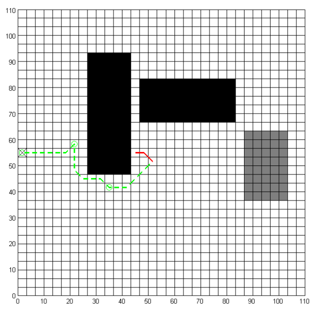
(c) Successful paths, $r_{sg} = 9$



(d) Local minima in steepest descent path, $r_{sg} = 3$

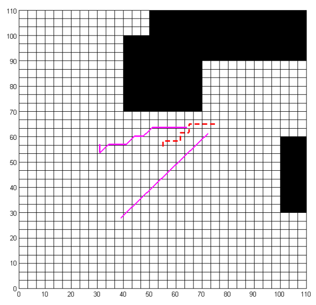


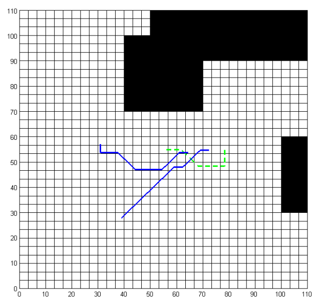
(e) Local minima in steepest descent path, $r_{sg} = 6$

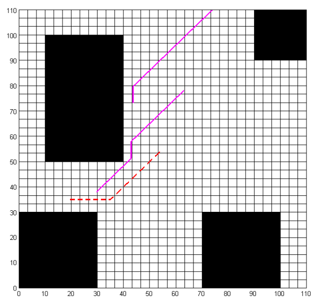


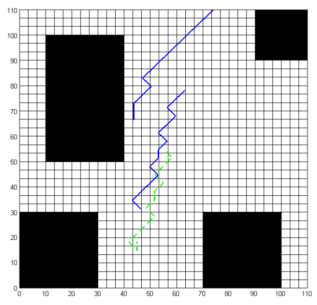
(f) Local minima in steepest descent path, $r_{sg} = 9$

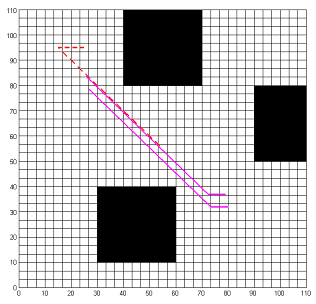
Figure C.3: Computed paths using steepest descent (solid red), and updated subgoals and steepest descent (dashed green), with subgoals as circles, and wave-front potential functions; sensed obstacles in black, sensed pursuers in grey, and the LG as X.


(a) Steepest descent path


(b) Subgoal based path


(c) Steepest descent path


(d) Subgoal based path


(e) Steepest descent path

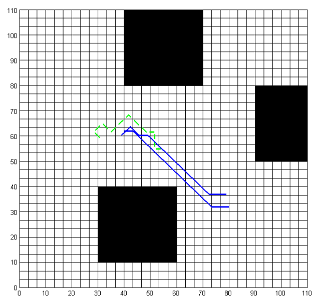
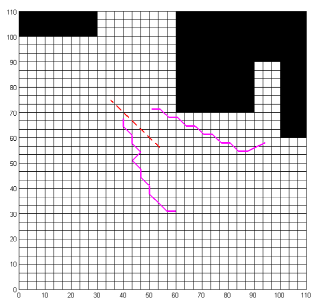
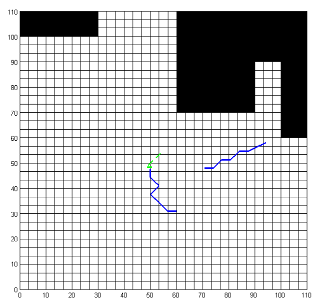
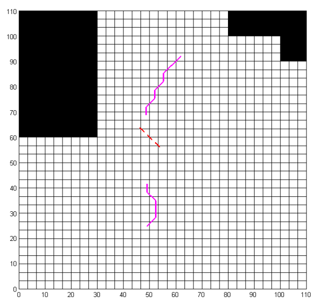
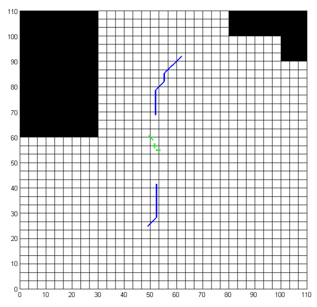

(f) Subgoal based path

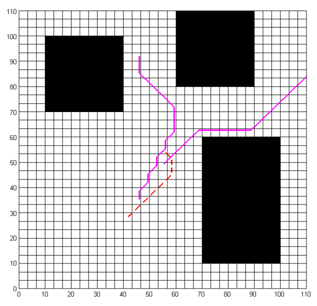
Figure C.4: Concentrated paths simulating the navigation of the robot and the pursuers with a velocity of $v_p = v_r$, and updating the sensing after every navigation motion, for a higher resolution grid.Steepest descent paths in red and corresponding pursuers in magenta; subgoal based paths in green and corresponding pursuers in blue; obstacles in black.


(a) Steepest descent path


(b) Subgoal based path


(c) Steepest descent path


(d) Subgoal based path


(e) Steepest descent path

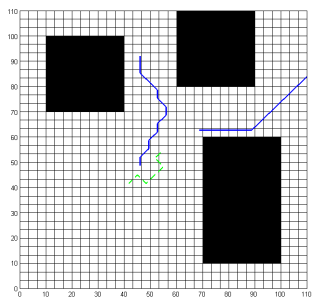
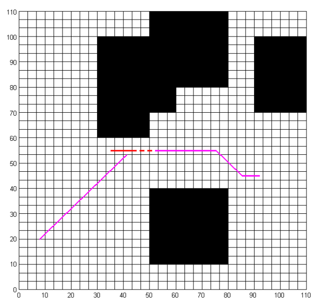
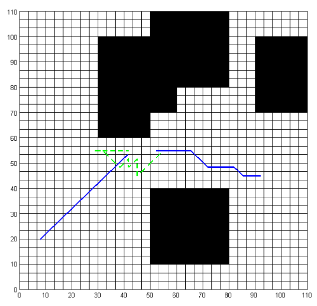
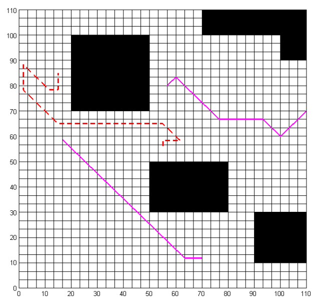
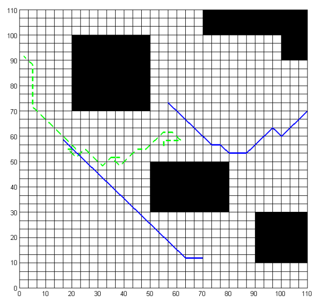

(f) Subgoal based path

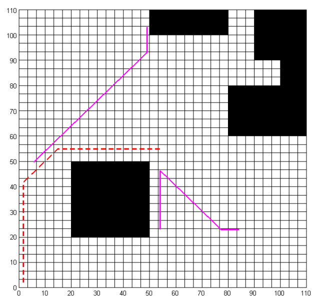
Figure C.5: Concentrated paths simulating the navigation of the robot and the pursuers with a velocity of $v_p = 2v_r$, and updating the sensing after every navigation motion, for a higher resolution grid. Steepest descent paths in red and corresponding pursuers in magenta; subgoal based paths in green and corresponding pursuers in blue; obstacles in black.


(a) Steepest descent path


(b) Subgoal based path


(c) Steepest descent path


(d) Subgoal based path


(e) Steepest descent path

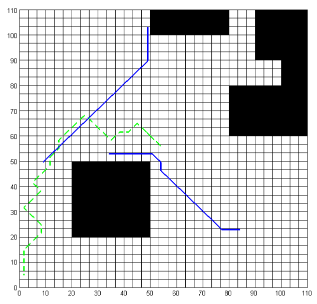
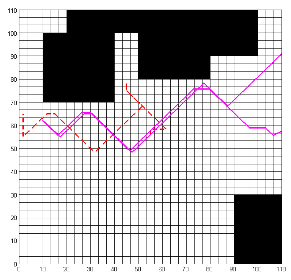
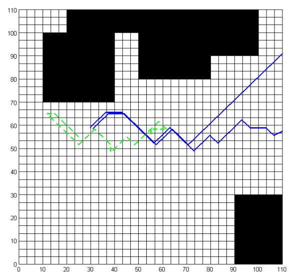
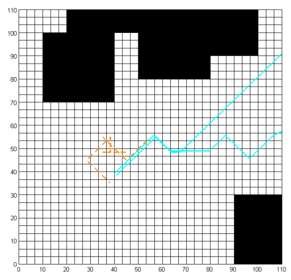
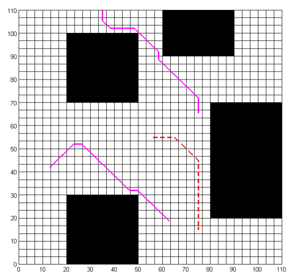

(f) Subgoal based path

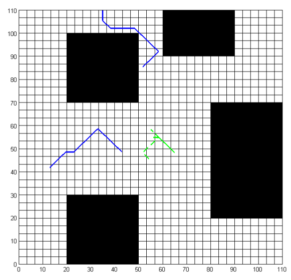
Figure C.6: Concentrated paths simulating the navigation of the robot and the pursuers with a velocity of $v_p = \frac{1}{2}v_r$, and updating the sensing after every navigation motion, for a higher resolution grid. Steepest descent paths in red and corresponding pursuers in magenta; subgoal based paths in green and corresponding pursuers in blue; obstacles in black.

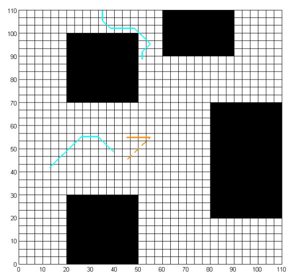

(a) Steepest descent path

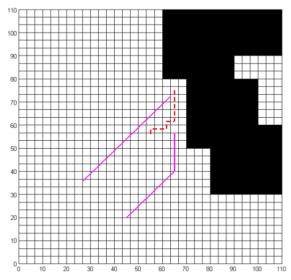

(b) Subgoal based path

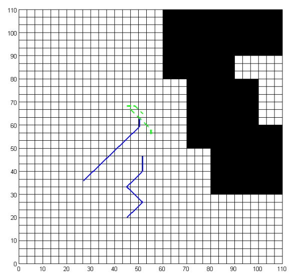

(c) General protean path


(d) Steepest descent path


(e) Subgoal based path


(f) General protean path


(g) Steepest descent path


(h) Subgoal based path

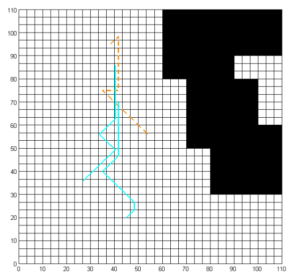
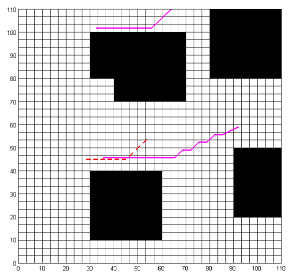
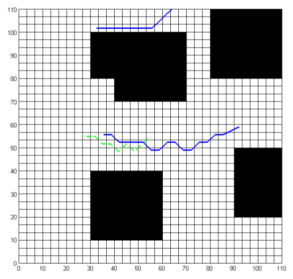
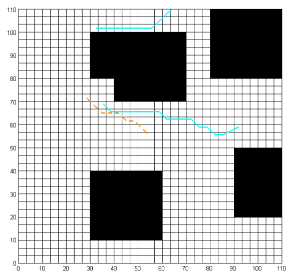
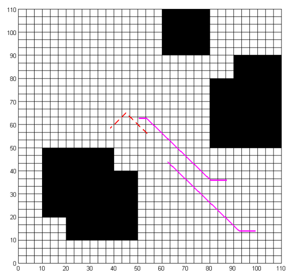

(i) General protean path

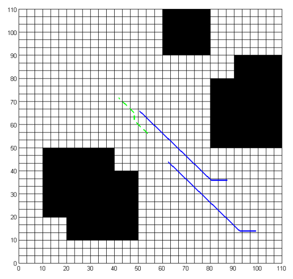
Figure C.7: Concentrated paths (protean general and guided, steepest descent) simulating the navigation of the robot and the pursuers with $v_p = v_r$, and updating the sensing after 3 navigation steps, for a higher resolution grid. Steepest descent paths in red and corresponding pursuers in magenta; subgoal based paths in green and corresponding pursuers in blue; general protean paths in orange and corresponding pursuers in cyan; obstacles in black.

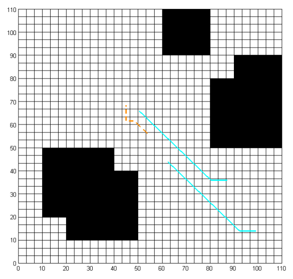

(a) Steepest descent path

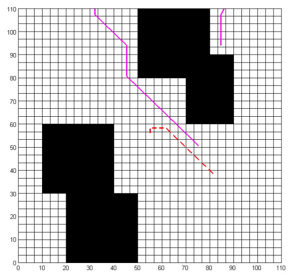

(b) Subgoal based path

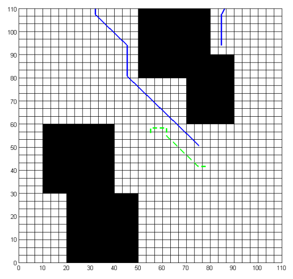

(c) General protean path


(d) Steepest descent path


(e) Subgoal based path


(f) General protean path


(g) Steepest descent path


(h) Subgoal based path

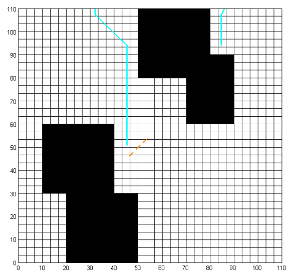
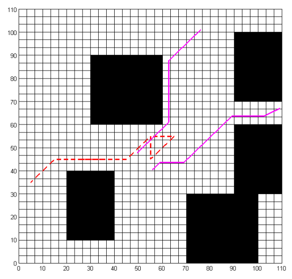
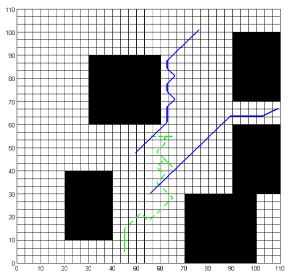
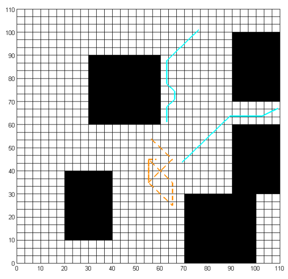
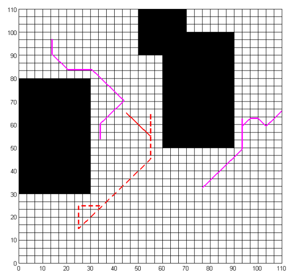

(i) General protean path

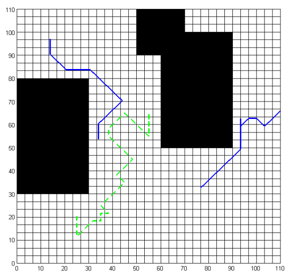
Figure C.8: Concentrated paths (protean general and guided, steepest descent) simulating the navigation of the robot and the pursuers with $v_p = 2v_r$, and updating the sensing after 3 navigation steps, for a higher resolution grid. Steepest descent paths in red and corresponding pursuers in magenta; subgoal based paths in green and corresponding pursuers in blue; general protean paths in orange and corresponding pursuers in cyan; obstacles in black.

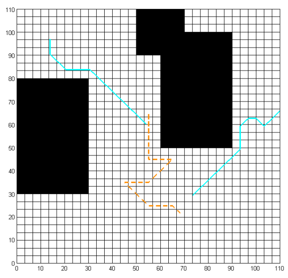

(a) Steepest descent path

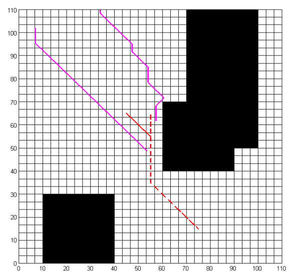

(b) Subgoal based path

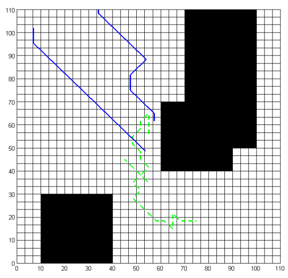

(c) General protean path


(d) Steepest descent path


(e) Subgoal based path


(f) General protean path


(g) Steepest descent path


(h) Subgoal based path

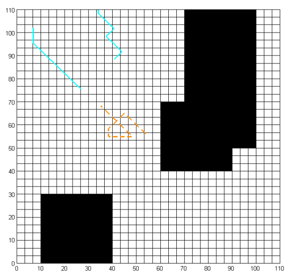

(i) General protean path

Figure C.9: Concentrated paths (protean general and guided, steepest descent) simulating the navigation of the robot and the pursuers with $v_p = \frac{1}{2}v_r$, and updating the sensing after 3 navigation steps, for a higher resolution grid. Steepest descent paths in red and corresponding pursuers in magenta; subgoal based paths in green and corresponding pursuers in blue; general protean paths in orange and corresponding pursuers in cyan; obstacles in black.

Appendix D

Tables of resulting DFD and RMSE to compare computed paths

This appendix presents the rest of the tables of resulting DFD and RMSE for low and high resolution grids, corresponding to Subsections 5.2.2 and 5.4.2.

For low resolution grids, Table D.1 presents the statistics for the modified FIRAS and quadratic potential functions. Tables D.2 and D.3 present the results of a BVP for Laplace equation with Jacobi and SOR iterative methods, respectively. Tables D.4 and D.5 continue with the results of a BVP for the perturbation equation with Jacobi and SOR iterative methods for 2, 5 and 10 pursuers, as the results are quite similar to the ones from the Laplace equation. Table D.6 presents the statistics for the computed paths with wave-front potential functions.

For high resolution grids, Table D.7 presents the statistics for the modified FIRAS and quadratic potential functions. Tables D.8 and D.9 present the results of a BVP for Laplace equation with Jacobi and SOR iterative methods, respectively, for 2, 5 and 10 pursuers. Tables D.10 and D.11 continue with the results of a BVP for the perturbation equation with Jacobi and SOR iterative methods for 2, 5 and 10 pursuers. Table D.12 presents the statistics for the computed paths with wave-front potential functions. Only some of the statistics are presented to illustrate the evaluation of the performance for all potential functions, as previous results show that using a BVP for the Laplace and perturbation equations does not produce the best results.

Table D.1: DFD and RMSE for modified FIRAS and quadratic potential functions.

PURSUERS		r_{sg} AND SUBGOAL COMMUTATION									
		EXTENDED SENSING					UNEXTENDED SENSING				
		2	3	4	5	6	2	3	4	5	6
1	μ DFD	23.404	24.284	24.207	15.298	0.132	23.814	24.692	24.096	16.101	1.259
	Sd DFD	24.698	27.584	27.953	23.120	2.493	25.242	27.888	25.909	24.020	7.243
	μ RMSE	7.119	7.432	7.692	5.205	0.034	8.303	8.442	8.568	5.620	0.508
	Sd RMSE	8.427	9.091	9.734	8.369	0.769	9.367	10.195	10.191	9.161	3.303
2	μ DFD	23.729	24.615	23.626	23.573	24.261	23.760	24.518	23.906	24.584	23.651
	Sd DFD	19.341	19.395	19.249	18.679	19.843	18.346	18.447	18.122	18.013	18.429
	μ RMSE	7.004	6.977	6.656	7.461	7.326	8.071	8.091	7.925	8.106	8.060
	Sd RMSE	6.781	6.776	6.609	7.266	7.232	7.178	7.305	6.952	7.150	7.353
3	μ DFD	23.943	25.395	25.398	20.681	10.838	24.729	24.989	25.152	19.864	10.868
	Sd DFD	17.342	18.800	18.843	21.565	20.720	16.472	17.791	18.958	21.373	20.826
	μ RMSE	7.031	8.126	8.696	6.849	3.855	7.999	8.223	8.210	6.583	3.909
	Sd RMSE	6.563	7.554	8.028	8.743	8.306	6.642	7.270	7.880	8.263	8.248
4	μ DFD	24.971	25.140	25.984	21.944	14.161	25.767	25.675	25.326	12.558	20.822
	Sd DFD	16.326	17.509	18.730	21.001	22.620	16.868	16.453	18.314	21.443	20.777
	μ RMSE	7.186	7.974	8.156	7.069	4.760	7.769	8.235	8.368	3.934	6.946
	Sd RMSE	6.175	7.148	7.591	8.095	8.913	6.368	7.026	7.749	8.120	8.350
5	μ DFD	24.734	25.590	26.229	19.540	12.163	25.270	25.855	25.336	19.142	11.617
	Sd DFD	16.461	17.534	18.941	20.549	20.962	16.068	17.208	19.050	20.800	20.550
	μ RMSE	6.858	7.898	8.305	6.463	4.259	7.683	8.081	8.112	6.314	3.987
	Sd RMSE	6.165	7.211	7.883	8.036	8.276	6.291	6.883	7.732	8.112	8.117
10	μ DFD	24.423	26.410	26.630	27.193	21.108	28.514	27.964	28.490	28.903	21.857
	Sd DFD	16.428	18.233	19.351	21.343	23.673	17.764	17.085	19.241	22.601	24.399
	μ RMSE	5.942	7.027	6.803	6.920	5.962	7.410	7.225	7.524	6.653	5.896
	Sd RMSE	5.787	7.028	7.469	7.808	8.846	6.396	6.481	7.411	7.752	8.764

Table D.2: DFD and RMSE for BVP for Laplace equation with Dirichlet boundary conditions and Jacobi iterative method.

PURSUERS		r_{sg} AND SUBGOAL COMMUTATION									
		EXTENDED SENSING					UNEXTENDED SENSING				
		2	3	4	5	6	2	3	4	5	6
1	μ DFD	42.126	28.719	18.457	4.8928	0	40.024	23.847	13.379	5.070	0.220
	Sd DFD	50.155	41.460	32.086	16.392	0	46.468	36.675	28.695	16.890	4.300
	μ RMSE	8.930	4.111	2.021	0.330	0	8.434	3.061	1.906	0.279	0.015
	Sd RMSE	12.898	8.122	5.398	1.853	0	12.459	7.025	5.667	1.834	0.220
2	μ DFD	42.751	28.275	18.122	7.289	2.082	41.045	23.675	14.068	4.182	1.476
	Sd DFD	37.353	34.991	29.280	20.741	12.707	37.650	32.748	27.782	15.370	10.679
	μ RMSE	8.777	4.338	2.012	0.598	0.061	7.913	3.087	1.307	0.319	0.038
	Sd RMSE	11.448	8.152	4.949	4.118	0.551	11.525	6.872	4.084	2.268	0.443
3	μ DFD	42.161	31.029	18.952	8.256	2.735	40.425	23.979	2.342	5.668	1.707
	Sd DFD	33.840	33.284	29.279	20.477	13.211	32.712	32.257	14.172	18.919	10.983
	μ RMSE	8.064	3.670	1.473	0.394	0.1064	7.914	3.328	0.071	0.474	0.088
	Sd RMSE	10.314	7.228	3.766	1.800	1.005	11.328	7.558	0767	3.095	0.986
4	μ DFD	41.252	29.019	20.953	11.026	4.772	41.687	25.210	15.353	6.379	1.887
	Sd DFD	30.902	32.138	30.305	25.001	18.627	31.998	33.288	27.791	20.421	11.369
	μ RMSE	8.450	2.825	1.841	0.405	0.106	7.491	2.662	1.371	0.281	0.098
	Sd RMSE	10.807	5.641	5.236	1.803	0.916	11.018	6.688	4.042	1.995	0.926
5	μ DFD	42.470	28.544	18.542	9.439	2.754	40.330	24.056	13.940	5.926	2.821
	Sd DFD	30.846	32.617	29.805	23.027	13.627	30.755	31.389	27.273	19.086	14.913
	μ RMSE	7.309	3.219	1.787	0.617	0.1761	6.936	3.072	0.979	0.440	0.087
	Sd RMSE	9.551	6.392	5.051	3.459	1.370	10.434	6.775	3.801	2.567	0.781
10	μ DFD	41.894	29.274	17.980	9.902	4.691	42.527	24.828	14.180	7.915	3.600
	Sd DFD	27.557	31.015	27.407	23.261	17.392	29.606	32.820	27.797	23.093	16.998
	μ RMSE	5.563	1.690	0.905	0.462	0.068	4.763	1.878	0.904	0.459	0.069
	Sd RMSE	7.542	4.338	2.857	2.616	0.709	9.062	5.637	3.682	2.935	0.722

Table D.3: DFD and RMSE for BVP for Laplace equation with Dirichlet boundary conditions and SOR iterative method.

PURSUERS		r_{sg} AND SUBGOAL COMMUTATION									
		EXTENDED SENSING					UNEXTENDED SENSING				
		2	3	4	5	6	2	3	4	5	6
1	μ DFD	31.140	6.060	3.968	1.361	0.027	29.448	6.006	3.989	1.575	0.064
	Sd DFD	41.104	14.684	11.289	6.300	0.595	38.538	15.472	10.808	5.671	0.832
	μ RMSE	8.950	1.199	0.300	0.040	0	7.832	1.293	0.297	0.061	0
	Sd RMSE	12.691	3.247	1.375	0.758	0	11.808	4.300	1.635	0.527	0
2	μ DFD	29.512	5.623	3.712	1.715	0.188	30.102	5.372	3.286	1.443	0.108
	Sd DFD	31.542	13.479	11.731	6.537	1.668	32.501	13.010	9.721	6.860	1.055
	μ RMSE	7.192	1.164	0.283	0.083	0.025	6.588	1.044	0.272	0.097	0
	Sd RMSE	9.644	3.364	1.326	0.700	0.517	9.622	3.226	1.377	0.808	0
3	μ DFD	29.224	5.599	2.731	1.2634	0.288	26.403	4.212	2.331	1.176	0.211
	Sd DFD	28.458	13.956	7.731	4.912	1.922	28.275	11.979	7.084	5.194	1.556
	μ RMSE	7.698	1.076	0.344	0.051	0.007	6.847	0.846	0.339	0.061	0.007
	Sd RMSE	9.639	3.193	1.668	0.435	0.202	9.594	2.955	1.631	0.708	0.200
4	μ DFD	28.283	5.649	2.587	1.407	0.174	27.405	4.498	2.755	1.339	0.263
	Sd DFD	26.572	14.140	9.462	7.424	1.372	28.236	13.240	9.753	5.381	1.672
	μ RMSE	6.841	1.122	0.276	0.092	0.005	6.360	0.874	0.256	0.080	0
	Sd RMSE	8.773	3.444	1.370	0.834	0.136	9.120	2.868	1.434	0.845	0
5	μ DFD	25.287	4.230	2.052	1.193	0.248	25.564	4.247	2.339	0.919	0.217
	Sd DFD	25.550	11.472	7.920	6.515	3.493	25.615	11.512	9.211	4.331	1.631
	μ RMSE	6.491	0.908	0.292	0.076	0	6.026	0.934	0.284	0.030	0.014
	Sd RMSE	8.933	3.087	1.552	0.800	0	8.738	3.073	1.578	0.383	0.435
10	μ DFD	23.222	3.861	1.709	0.636	0.422	23.288	2.904	1.614	1.006	0.465
	Sd DFD	22.093	11.749	8.471	5.000	4.729	24.719	10.495	8.470	6.858	3.732
	μ RMSE	3.981	0.615	0.100	0.009	0.015	3.239	0.517	0.217	0.026	0.006
	Sd RMSE	6.511	2.500	0.0771	0.168	0.277	6.519	2.258	1.380	0.328	0.176

Table D.4: DFD and RMSE for BVP for perturbation equation with Dirichlet boundary conditions and Jacobi iterative method.

PURSUERS		r_{sg} AND SUBGOAL COMMUTATION									
		EXTENDED SENSING					UNEXTENDED SENSING				
		2	3	4	5	6	2	3	4	5	6
2	μ DFD	41.193	29.872	19.813	10.850	2.039	40.993	26.721	16.420	8.024	1.436
	Sd DFD	36.977	35.289	30.870	25.438	10.665	35.754	34.742	29.816	22.607	10.056
	μ RMSE	7.654	4.582	2.163	0.747	0.109	7.815	3.502	1.879	0.736	0.190
	Sd RMSE	9.980	8.135	5.744	3.445	0.885	11.403	7.611	5.810	3.751	2.277
5	μ DFD	41.775	31.607	21.298	9.972	4.306	41.567	25.806	15.812	7.945	4.005
	Sd DFD	30.650	32.984	29.907	23.074	16.250	31.663	32.591	27.932	22.189	16.217
	μ RMSE	7.583	3.396	1.844	0.716	0.296	7.204	3.198	1.790	0.520	0.415
	Sd RMSE	10.195	7.038	4.124	3.416	2.018	10.977	7.379	5.885	2.598	2.994
10	μ DFD	-	-	-	-	-	46.449	31.316	22.7024	15.497	12.639
	Sd DFD	-	-	-	-	-	29.725	33.829	31.458	27.073	26.672
	μ RMSE	-	-	-	-	-	5.985	2.570	2.095	1.583	1.450
	Sd RMSE	-	-	-	-	-	9.991	6.374	6.321	4.856	5.087

Table D.5: DFD and RMSE forBVP for perturbation equation with Dirichlet boundary conditions and SOR iterative method.

PURSUERS		r_{sg} AND SUBGOAL COMMUTATION									
		EXTENDED SENSING					UNEXTENDED SENSING				
		2	3	4	5	6	2	3	4	5	6
2	μ DFD	30.175	5.728	4.200	1.830	0.178	28.683	5.756	4.152	2.266	0.457
	Sd DFD	33.201	13.785	12.853	7.054	2.540	30.802	13.879	11.091	9.225	3.019
	μ RMSE	8.039	1.192	0.466	0.152	0.009	7.353	1.183	0.473	0.113	0.018
	Sd RMSE	10.397	3.515	2.589	1.047	0.250	10.205	3.608	2.129	1.107	0.356
5	μ DFD	27.609	5.971	2.848	1.680	0.973	27.861	6.526	3.711	1.649	0.557
	Sd DFD	27.321	14.211	8.532	6.584	7.138	27.890	15.871	11.426	7.755	4.254
	μ RMSE	6.640	1.192	0.400	0.206	0.137	6.194	1.218	0.461	0.148	0.026
	Sd RMSE	9.164	3.751	1.727	1.890	1.350	9.109	3.778	2.284	1.342	0.483
10	μ DFD	-	-	-	-	-	25.619	6.772	4.428	3.344	2.533
	Sd DFD	-	-	-	-	-	29.606	17.656	14.157	11.798	11.452
	μ RMSE	-	-	-	-	-	3.571	0.939	0.723	0.480	0.254
	Sd RMSE	-	-	-	-	-	7.012	3.582	2.980	2.386	1.567

Table D.6: DFD and RMSE for wave-front potential functions

PURSUERS		r_{sg} AND SUBGOAL COMMUTATION									
		EXTENDED SENSING					UNEXTENDED SENSING				
		2	3	4	5	6	2	3	4	5	6
1	μ DFD	26.083	23.864	24.019	14.996	0.469	28.320	26.196	23.430	13.372	3.109
	Sd DFD	29.575	26.931	27.572	22.972	6.388	31.560	29.490	28.654	22.198	13.154
	μ RMSE	7.432	7.383	7.542	4.750	0.092	8.786	8.678	7.928	4.668	0.987
	Sd RMSE	9.025	9.198	9.951	7.969	1.162	10.397	10.525	10.656	8.480	4.692
2	μ DFD	31.719	30.807	28.936	20.999	10.752	32.869	29.666	28.126	20.897	9.496
	Sd DFD	26.745	26.272	27.013	26.511	24.567	26.270	25.942	25.192	26.053	22.134
	μ RMSE	8.538	8.498	8.765	6.725	2.964	9.739	9.296	9.275	6.700	2.603
	Sd RMSE	8.262	8.631	9.724	9.764	7.884	8.843	9.538	9.573	9.712	7.373
3	μ DFD	35.244	33.976	31.655	26.625	16.355	34.787	34.049	31.288	25.269	15.655
	Sd DFD	25.018	26.134	25.834	28.240	27.455	23.226	25.234	24.978	26.960	27.125
	μ RMSE	9.779	9.592	9.556	8.061	4.488	9.835	9.624	9.280	7.685	4.182
	Sd RMSE	8.724	9.140	9.852	9.698	9.283	8.308	8.954	9.392	9.772	9.030
4	μ DFD	36.559	36.835	33.142	27.894	20.467	38.173	35.151	34.417	27.371	19.076
	Sd DFD	23.235	25.070	24.544	26.302	29.060	23.411	23.988	24.961	27.736	28.151
	μ RMSE	8.919	9.882	9.359	8.158	5.201	10.365	10.020	9.704	7.810	4.464
	Sd RMSE	8.106	8.826	9.087	9.439	9.349	8.496	8.773	9.210	9.555	8.741
5	μ DFD	34.028	34.073	30.428	23.722	15.398	37.753	32.853	32.489	23.826	18.911
	Sd DFD	21.807	23.791	23.953	25.369	26.298	22.343	22.537	23.750	26.191	29.547
	μ RMSE	9.256	8.922	8.774	7.483	4.436	10.218	9.508	9.236	6.954	4.913
	Sd RMSE	8.124	8.420	9.026	9.576	9.223	8.164	8.862	9.134	9.051	9.567
10	μ DFD	42.462	42.389	40.082	36.819	33.697	43.524	42.059	41.555	37.488	33.940
	Sd DFD	23.676	24.249	25.086	25.061	29.221	23.005	24.037	24.425	26.798	30.762
	μ RMSE	9.996	9.484	9.895	9.717	8.428	10.821	10.456	10.557	9.544	7.852
	Sd RMSE	8.142	8.505	9.070	9.180	10.228	8.291	8.691	9.194	9.691	10.532

Table D.7: DFD and RMSE for modified FIRAS and quadratic potential functions.

PURSUERS		r_{sg} AND SUBGOAL COMMUTATION					
		3	6	9	12	15	18
1	μ DFD	16.877	26.743	25.617	27.221	24.639	14.365
	Sd DFD	17.318	32.112	31.439	28.442	29.849	25.589
	μ RMSE	6.310	7.188	6.813	7.504	7.291	4.550
	Sd RMSE	6.543	9.357	9.282	8.842	9.432	8.974
2	μ DFD	21.520	22.689	24.965	26.054	24.272	18.217
	Sd DFD	20.202	19.118	18.584	24.495	23.926	26.519
	μ RMSE	6.123	6.995	8.080	7.432	7.016	4.811
	Sd RMSE	5.357	6.058	7.555	8.809	9.277	8.782
3	μ DFD	20.692	23.983	28.291	27.206	26.537	21.361
	Sd DFD	14.019	17.449	19.159	18.990	22.308	25.802
	μ RMSE	6.427	7.423	8.064	9.144	7.461	5.536
	Sd RMSE	4.971	6.820	7.357	8.085	8.898	8.417
4	μ DFD	21.828	29.265	26.862	26.535	26.459	19.408
	Sd DFD	17.762	20.766	17.357	19.026	20.675	22.302
	μ RMSE	6.073	7.775	7.051	7.950	8.156	5.734
	Sd RMSE	4.401	6.131	6.414	7.744	8.711	8.726
5	μ DFD	21.193	26.177	25.658	25.240	21.377	22.288
	Sd DFD	12.765	17.806	17.317	20.981	20.984	25.362
	μ RMSE	6.091	7.146	8.138	6.441	6.178	6.508
	Sd RMSE	4.140	5.792	7.315	7.136	8.276	9.043
10	μ DFD	19.908	26.217	28.770	29.347	27.962	33.318
	Sd DFD	16.260	19.270	22.456	19.767	22.296	25.671
	μ RMSE	4.810	7.613	6.231	7.486	6.640	7.680
	Sd RMSE	4.542	6.219	7.346	7.617	8.114	10.034

Table D.8: DFD and RMSE for BVP for Laplace equation with Dirichlet boundary conditions and Jacobi iterative method.

PURSUERS		r_{sg} AND SUBGOAL COMMUTATION					
		3	6	9	12	15	18
2	μ DFD	2.616	1.625	0.321	0.506	0.847	0.637
	Sd DFD	7.863	5.824	1.316	2.316	3.735	2.557
	μ RMSE	1.147	0.265	0.068	0.009	0	0
	Sd RMSE	2.020	0.849	0.301	0.051	0	0
5	μ DFD	2.729	0.816	0.438	0.190	0.036	0.076
	Sd DFD	7.427	3.479	1.720	0.944	0.340	0.487
	μ RMSE	1.262	0.286	0.122	0.035	0	0
	Sd RMSE	2.420	1.069	0.486	0.237	0	0
10	μ DFD	2.565	0.557	0.101	0.144	0.469	0.468
	Sd DFD	8.620	1.780	0.571	0.811	0.469	0.468
	μ RMSE	0.786	0.223	0.028	0.015	0	0
	Sd RMSE	2.417	0.942	0.195	0.146	0.047	0.067

Table D.9: DFD and RMSE for BVP for Laplace equation with Dirichlet boundary conditions and SOR iterative method.

		r_{sg} AND SUBGOAL COMMUTATION					
PURSUERS		3	6	9	12	15	18
2	μ DFD	3.801	1.254	1.514	1.176	0.047	0.897
	Sd DFD	9.190	3.593	5.092	5.801	0.369	3.682
	μ RMSE	1.265	0.462	0.114	0.010	0	0
	Sd RMSE	2.314	1.190	0.504	0.066	0	0
5	μ DFD	2.755	0.869	0.303	0.199	0.858	0.036
	Sd DFD	9.317	4.184	1.580	1.490	5.205	0.341
	μ RMSE	0.965	0.314	0.055	0	0	0
	Sd RMSE	2.325	1.553	0.323	0	0	0
10	μ DFD	1.387	1.051	0.261	0.198	0.081	0.181
	Sd DFD	2.373	5.362	1.505	1.143	0.572	0.181
	μ RMSE	0.564	0.353	0.055	0	0	0
	Sd RMSE	1.476	1.304	0.331	0	0	0

Table D.10: DFD and RMSE for BVP for Laplace with perturbation equation with Dirichlet boundary conditions and Jacobi iterative method.

		r_{sg} AND SUBGOAL COMMUTATION					
PURSUERS		3	6	9	12	15	18
2	μ DFD	2.942	1.337	0.960	0.667	0.604	1.225
	Sd DFD	6.356	5.535	2.870	5.053	3.960	7.901
	μ RMSE	1.354	0.171	0.183	0.015	0	0.082
	Sd RMSE	2.809	0.679	0.576	0.083	0	0.708
5	μ DFD	2.509	1.664	1.618	0.880	0.179	0.230
	Sd DFD	4.798	6.152	8.166	4.718	1.127	1.005
	μ RMSE	1.336	0.468	0.191	0.057	0.007	0.074
	Sd RMSE	2.815	2.072	0.720	0.346	0.063	0.710
10	μ DFD	2.273	0.516	0.200	0.411	0.483	0.724
	Sd DFD	5.721	1.358	0.921	1.929	1.979	3.863
	μ RMSE	0.795	0.282	0.085	0.131	0.080	0.017
	Sd RMSE	1.656	1.071	0.491	0.708	0.459	0.125

Table D.11: DFD and RMSE for BVP for Laplace with perturbation equation with Dirichlet boundary conditions and SOR iterative method.

		r_{sg} AND SUBGOAL COMMUTATION					
PURSUERS		3	6	9	12	15	18
2	μ DFD	3.144	2.364	1.307	0.339	0.305	0.508
	Sd DFD	6.052	7.815	5.030	2.364	2.356	1.567
	μ RMSE	1.607	0.291	0.210	0	0.061	0.099
	Sd RMSE	3.469	0.870	0.869	0	0.531	0.804
5	μ DFD	3.381	1.845	0.363	0.148	0.311	0.103
	Sd DFD	10.828	8.198	1.321	0.840	1.687	0.682
	μ RMSE	0.810	0.274	0.232	0.035	0	0
	Sd RMSE	1.767	0.800	0.918	0.207	0	0
10	μ DFD	3.193	0.744	1.947	0.920	0	0.216
	Sd DFD	9.445	1.796	10.620	7.599	0	0.981
	μ RMSE	1.063	0.351	0.429	0.329	0.298	0.405
	Sd RMSE	2.086	1.158	1.420	1.412	1.054	1.205

Table D.12: DFD and RMSE for wave-front potential functions.

		r_{sg} AND SUBGOAL COMMUTATION					
PURSUERS		3	6	9	12	15	18
1	μ DFD	19.673	25.638	28.680	28.278	24.341	11.297
	Sd DFD	21.905	27.392	30.052	30.706	33.269	22.005
	μ RMSE	5.640	7.575	6.968	8.401	5.851	3.977
	Sd RMSE	6.381	9.223	8.877	10.126	9.625	10.569
2	μ DFD	31.631	35.999	40.994	35.345	39.793	19.203
	Sd DFD	28.661	29.180	36.495	33.820	30.226	31.074
	μ RMSE	8.741	9.100	9.505	9.289	11.520	5.286
	Sd RMSE	8.106	8.253	10.007	10.089	12.187	10.012
3	μ DFD	35.329	40.756	44.965	34.625	37.266	28.079
	Sd DFD	30.483	32.324	28.545	25.528	26.160	32.503
	μ RMSE	9.096	11.713	11.734	10.979	10.787	8.091
	Sd RMSE	6.919	9.985	9.115	10.434	11.553	11.278
4	μ DFD	48.460	42.119	44.658	41.893	38.284	32.801
	Sd DFD	33.267	27.860	28.480	28.970	30.147	33.638
	μ RMSE	11.732	9.299	9.756	9.820	10.565	9.199
	Sd RMSE	8.378	7.426	9.035	9.415	10.131	11.884
5	μ DFD	35.761	37.110	32.409	34.724	31.891	28.695
	Sd DFD	26.479	26.286	24.077	26.049	27..813	31.569
	μ RMSE	8.535	8.196	8.158	10.146	7.361	8.631
	Sd RMSE	6.448	7.461	7.461	7.938	10.442	12.027
10	μ DFD	47.066	50.504	54.591	51.277	46.336	43.263
	Sd DFD	31.458	28.936	25.353	27.689	27.269	29.052
	μ RMSE	10.018	10.134	12.084	8.898	10.312	7.948
	Sd RMSE	8.420	8.815	9.444	8.240	10.835	9.854

Appendix E

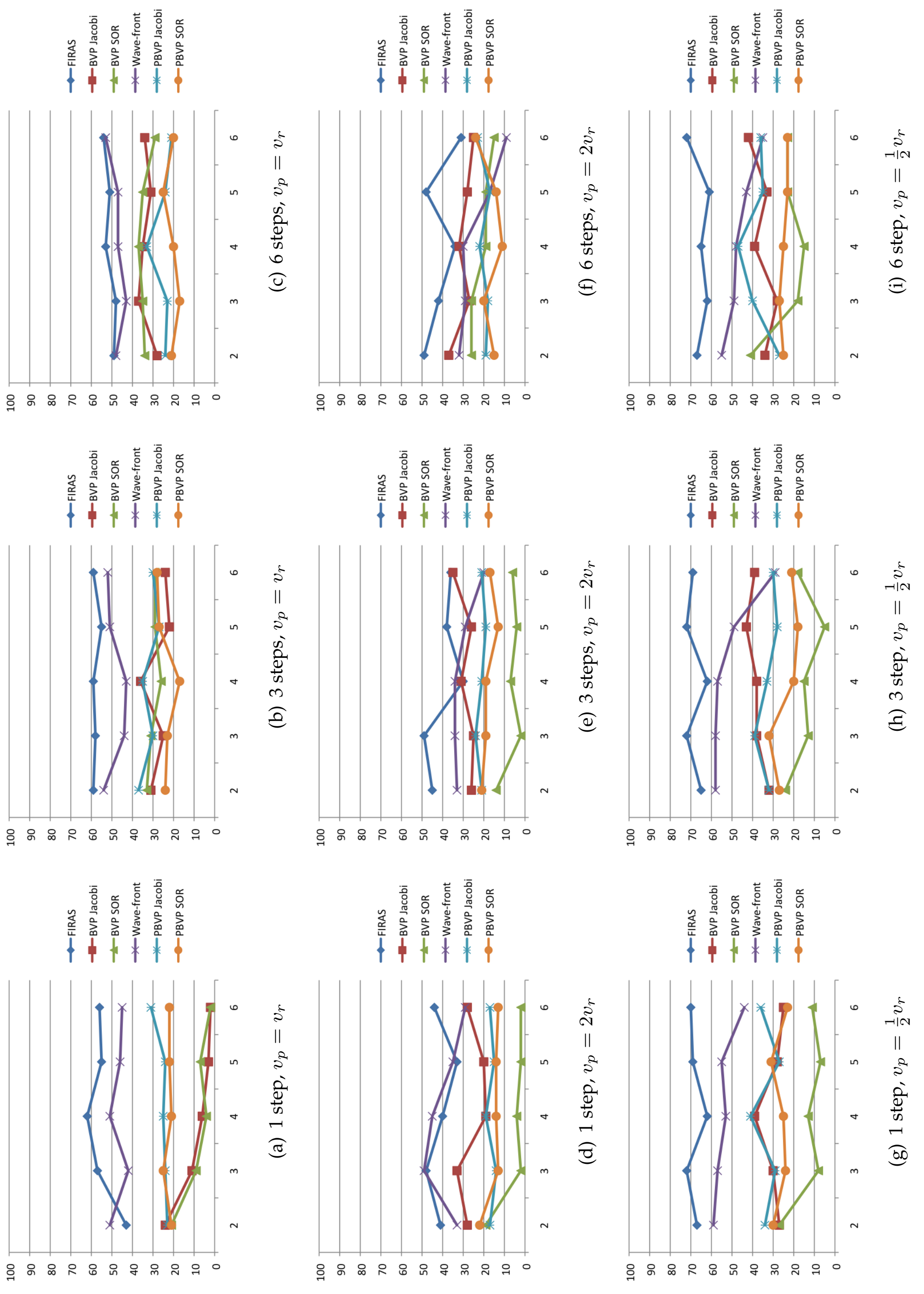
Additional statistical results

Figures E.1, E.2, E.3, E.4, E.5 and E.6 show additional results on the success of avoiding capture for the computed paths and pursuers in different numbers and velocities, in a low resolution grid. Figures E.7, E.8 and E.9 show more box plots for the times of capture in terms of navigation steps, for the same simulations on success of the paths. These figures complement Subsection 5.3.2.

Posteriorly, Figures E.10, E.11, E.12, E.13, E.14 and E.15 show additional results on the success of avoiding capture for the computed paths and pursuers in different numbers and velocities, in a high resolution grid. Figures E.16, E.17 and E.18 show more box plots for the times of capture in terms of navigation steps, for the same simulations on success of the paths. These figures complement Subsection 5.5.2.

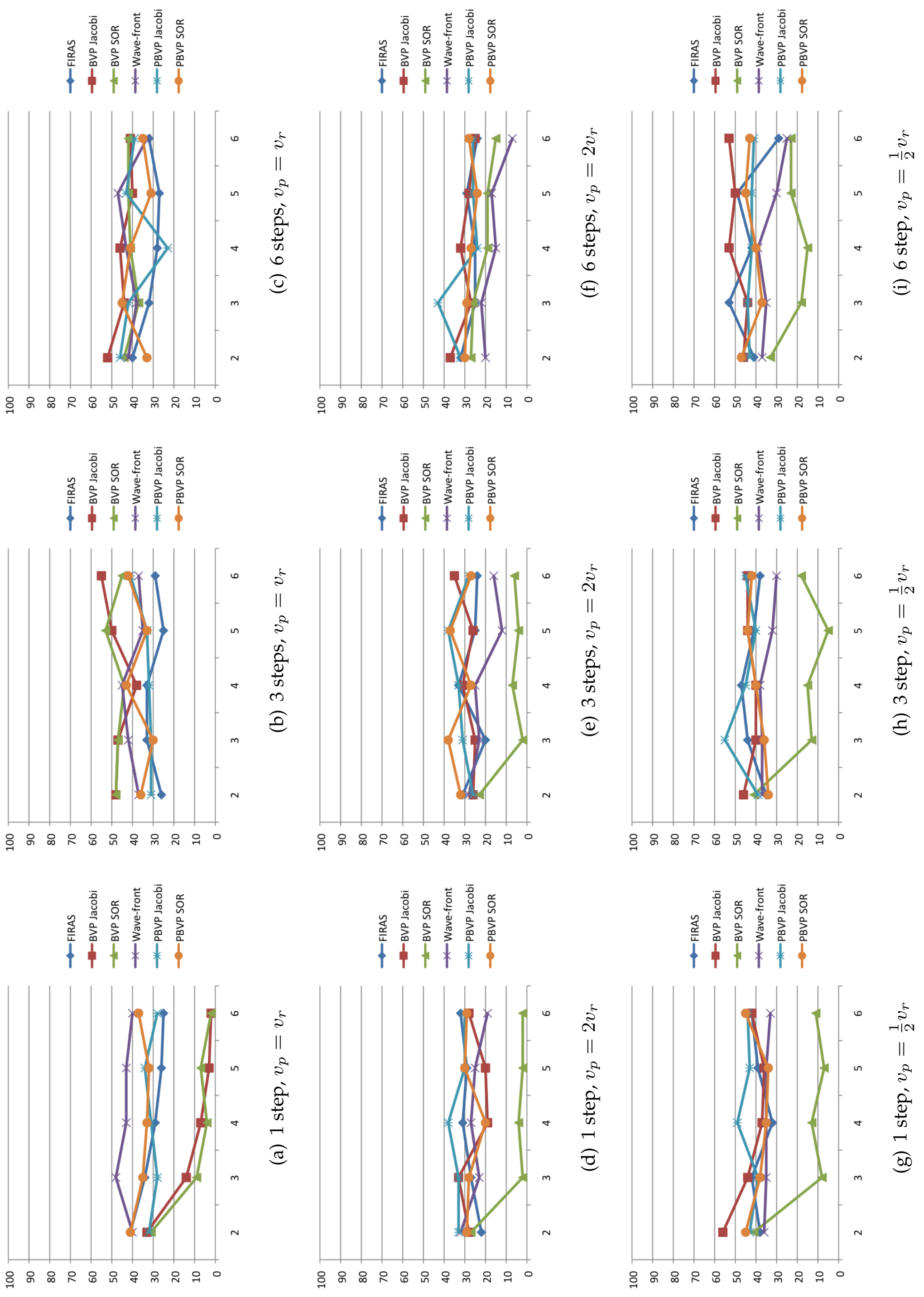
Figures E.19, E.20 and E.21 show additional results on the success of avoiding capture for different kinds of paths and pursuers in different numbers and velocities, in a high resolution grid. Figures E.22, E.23 and E.24 show more box plots for the times of capture in terms of navigation steps, for the same simulations on success of the paths. These figures complement Subsection 5.6.2.

Finally, Figure E.25 shows additional results on the success in avoiding capture for the reinforcement learning strategies, for the two proposed reward functions, and pursuers in different numbers (2, 4 and 5) and velocities. Figure E.26 shows the corresponding times of capture for the same numbers of pursuers.



(a) 1 step, $v_p = v_r$
(b) 3 steps, $v_p = v_r$
(c) 6 steps, $v_p = v_r$
(d) 1 step, $v_p = 2v_r$
(e) 3 steps, $v_p = 2v_r$
(f) 6 steps, $v_p = 2v_r$
(g) 1 step, $v_p = \frac{1}{2}v_r$
(h) 3 step, $v_p = \frac{1}{2}v_r$
(i) 6 step, $v_p = \frac{1}{2}v_r$

Figure E.1: Success of steepest descent paths for 2 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 1 step, $v_p = v_r$

(b) 3 steps, $v_p = v_r$

(c) 6 steps, $v_p = v_r$

(d) 1 step, $v_p = 2v_r$

(e) 3 steps, $v_p = 2v_r$

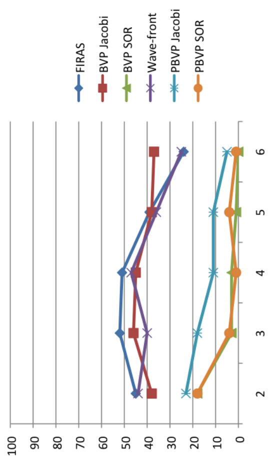
(f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$

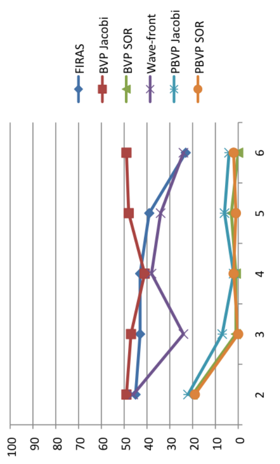
(h) 3 step, $v_p = \frac{1}{2}v_r$

(i) 6 step, $v_p = \frac{1}{2}v_r$

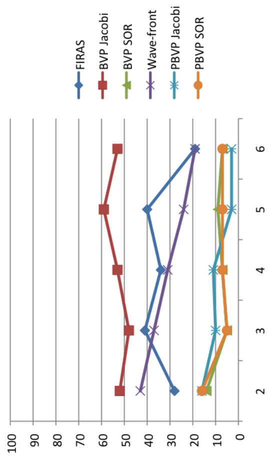
Figure E.2: Success of subgoal based paths for 2 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



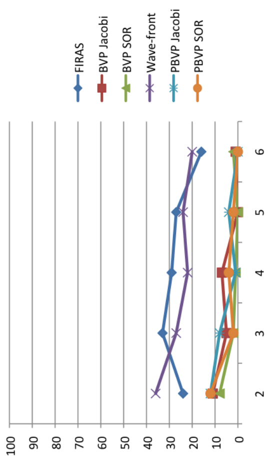
(a) 1 step, $v_p = v_r$



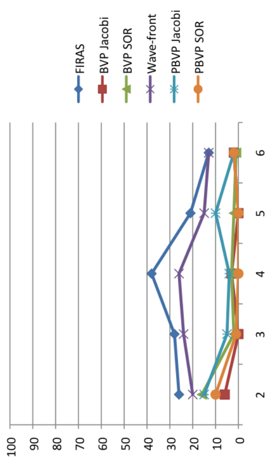
(b) 3 steps, $v_p = v_r$



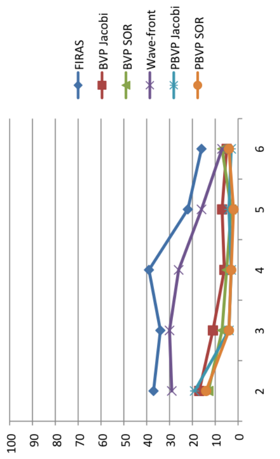
(c) 6 steps, $v_p = v_r$



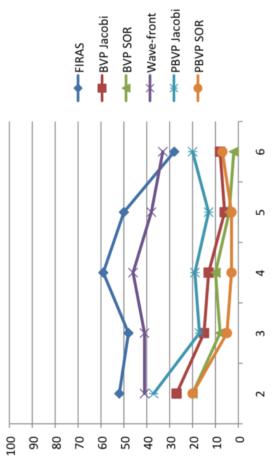
(d) 1 step, $v_p = 2v_r$



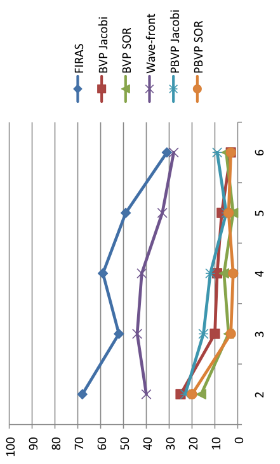
(e) 3 steps, $v_p = 2v_r$



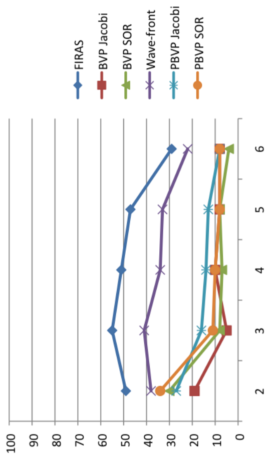
(f) 6 steps, $v_p = 2v_r$



(g) 1 step, $v_p = \frac{1}{2}v_r$

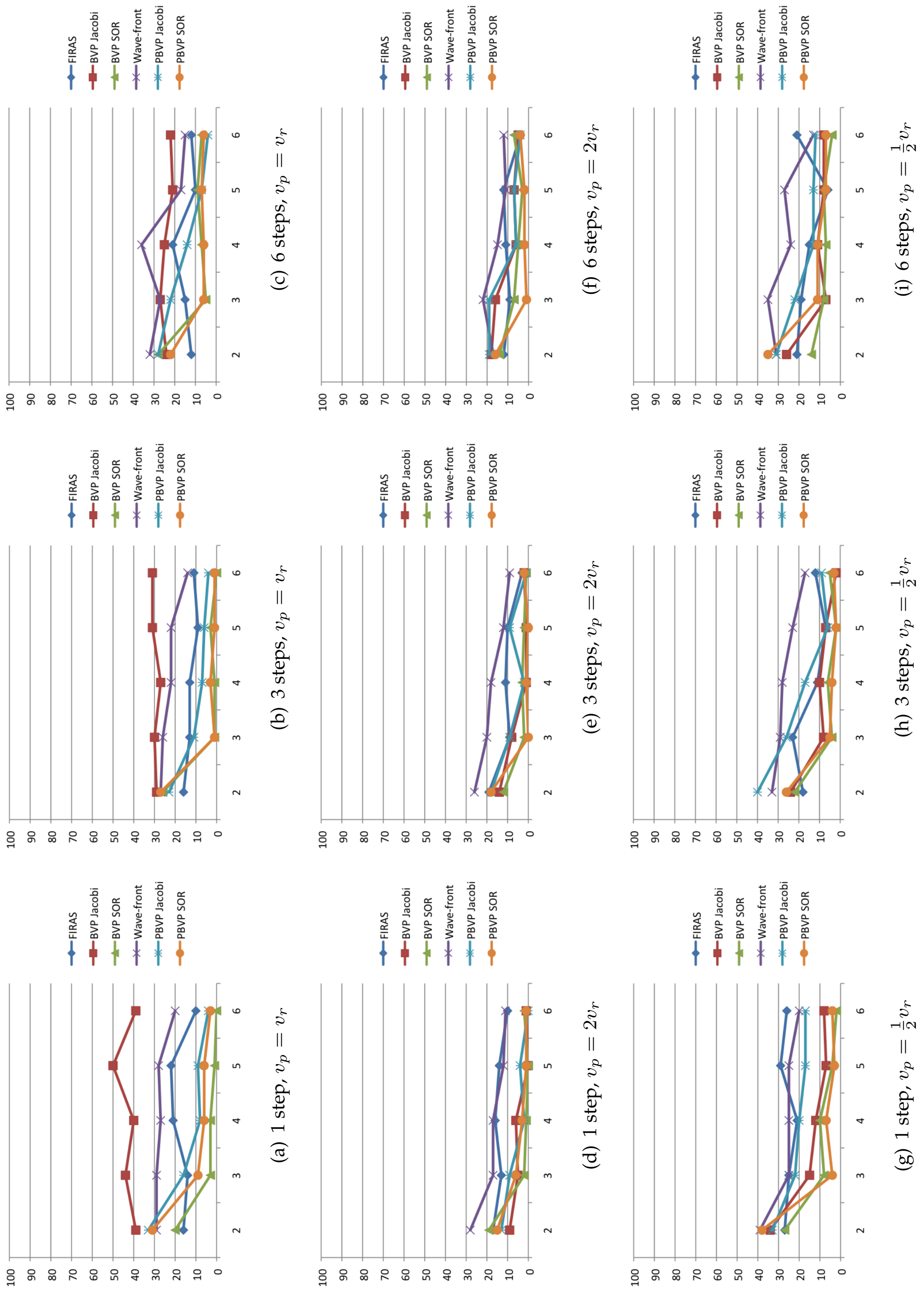


(h) 3 steps, $v_p = \frac{1}{2}v_r$



(i) 6 steps, $v_p = \frac{1}{2}v_r$

Figure E.3: Success of steepest descent paths for 4 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 1 step, $v_p = v_r$

(b) 3 steps, $v_p = v_r$

(c) 6 steps, $v_p = v_r$

(d) 1 step, $v_p = 2v_r$

(e) 3 steps, $v_p = 2v_r$

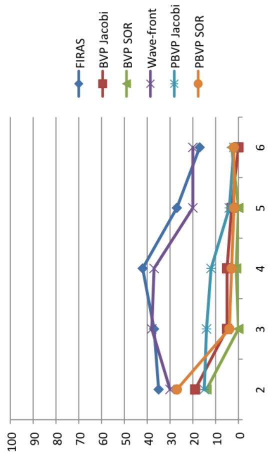
(f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$

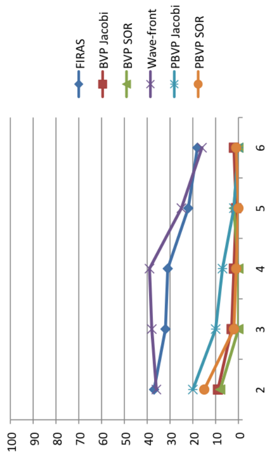
(h) 3 steps, $v_p = \frac{1}{2}v_r$

(i) 6 steps, $v_p = \frac{1}{2}v_r$

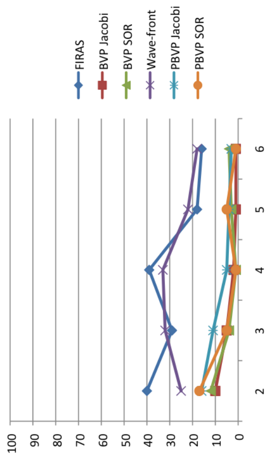
Figure E.4: Success of subgoal based paths for 4 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



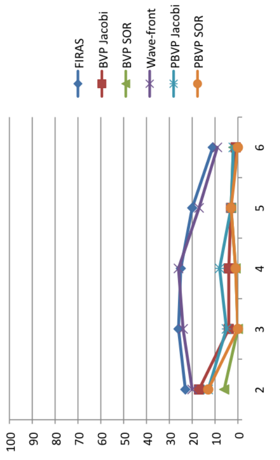
(a) 1 step, $v_p = v_r$



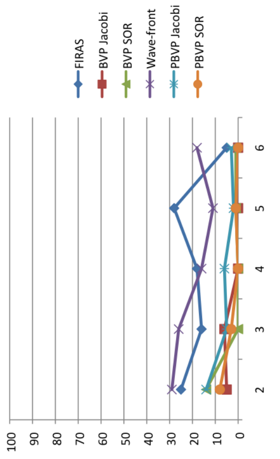
(b) 3 steps, $v_p = v_r$



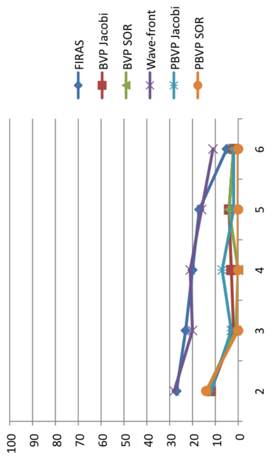
(c) 6 steps, $v_p = v_r$



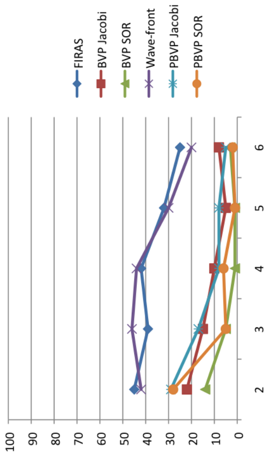
(d) 1 step, $v_p = 2v_r$



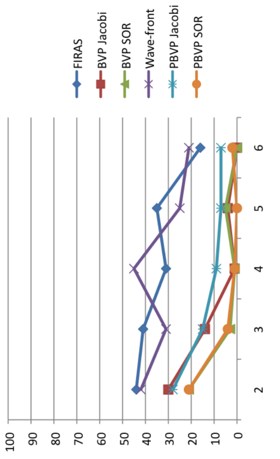
(e) 3 steps, $v_p = 2v_r$



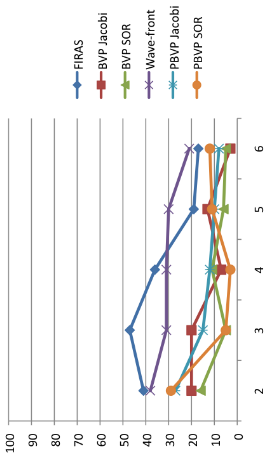
(f) 6 steps, $v_p = 2v_r$



(g) 1 step, $v_p = \frac{1}{2}v_r$

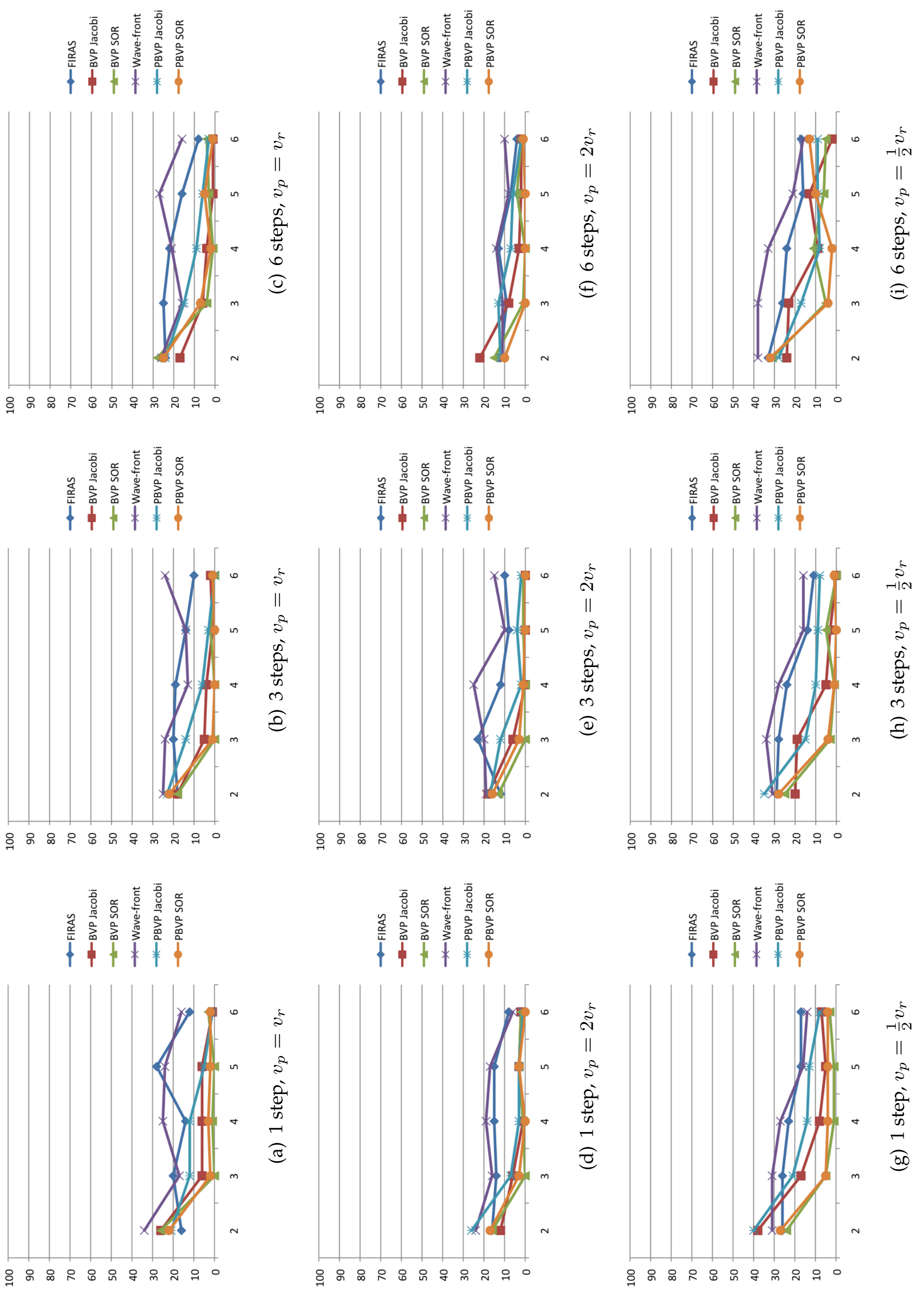


(h) 3 steps, $v_p = \frac{1}{2}v_r$



(i) 6 steps, $v_p = \frac{1}{2}v_r$

Figure E.5: Success of steepest descent paths for 5 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.

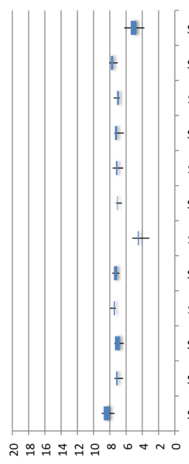


(a) 1 step, $v_p = v_r$ (b) 3 steps, $v_p = v_r$ (c) 6 steps, $v_p = v_r$

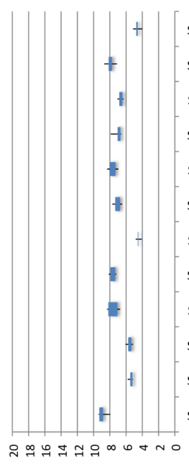
(d) 1 step, $v_p = 2v_r$ (e) 3 steps, $v_p = 2v_r$ (f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$ (h) 3 steps, $v_p = \frac{1}{2}v_r$ (i) 6 steps, $v_p = \frac{1}{2}v_r$

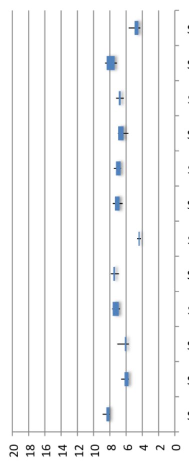
Figure E.6: Success of subgoal based paths for 5 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



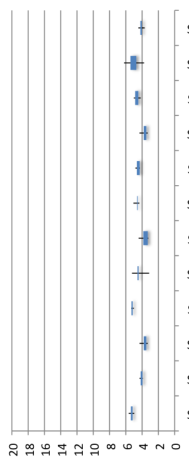
(a) 1 step, $v_p = v_r$



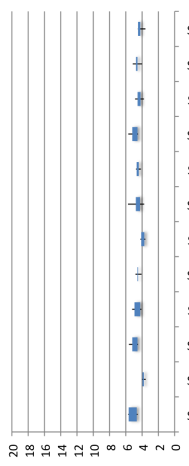
(b) 3 steps, $v_p = v_r$



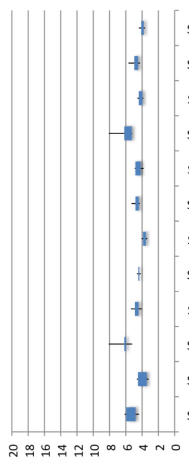
(c) 6 steps, $v_p = v_r$



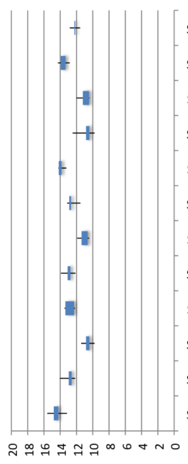
(d) 1 step, $v_p = 2v_r$



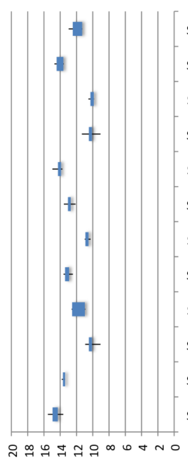
(e) 3 steps, $v_p = 2v_r$



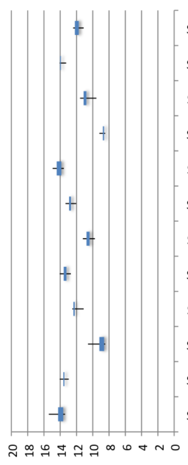
(f) 6 steps, $v_p = 2v_r$



(g) 1 step, $v_p = \frac{1}{2}v_r$

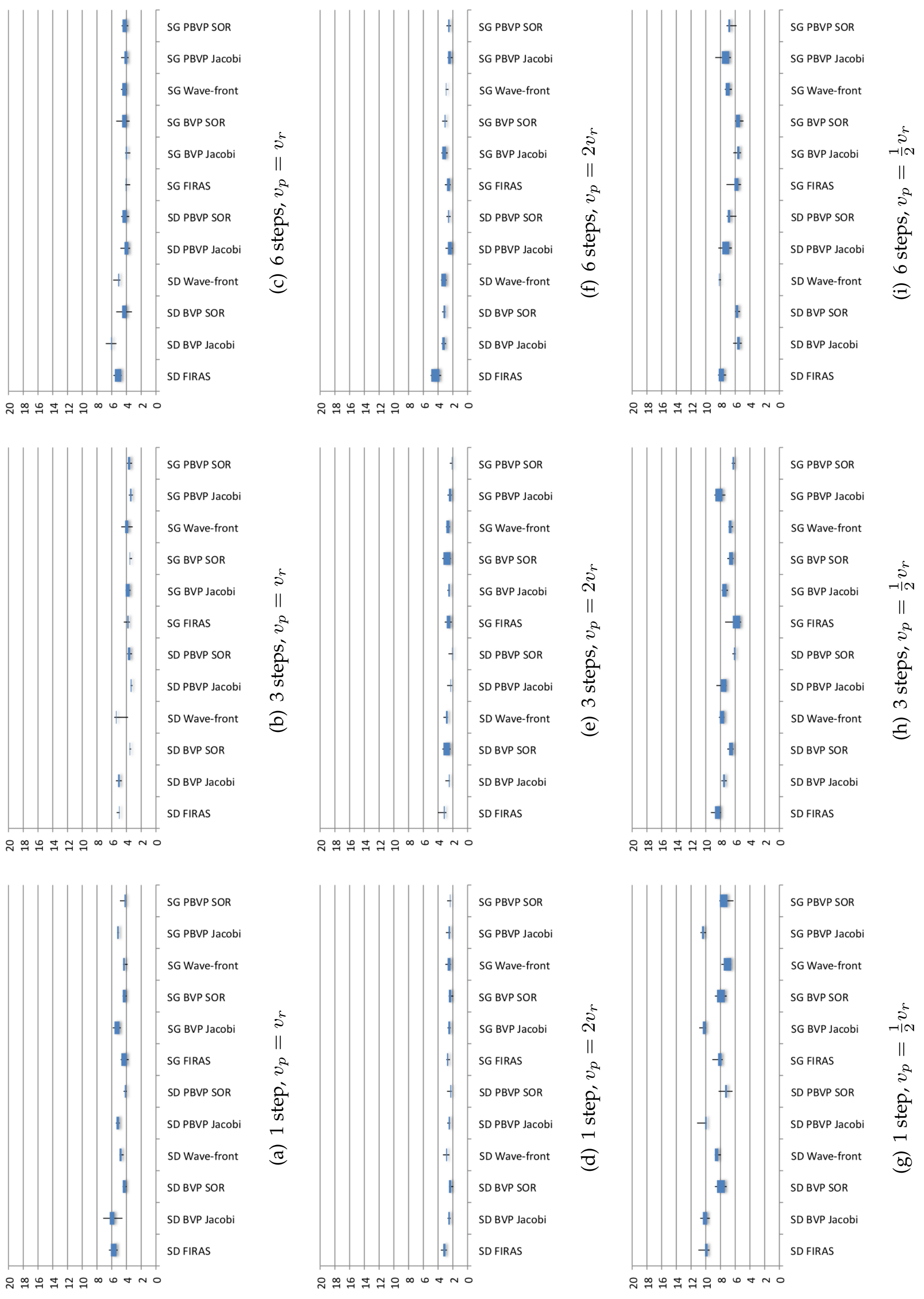


(h) 3 steps, $v_p = \frac{1}{2}v_r$



(i) 6 steps, $v_p = \frac{1}{2}v_r$

Figure E.7: Capture times (in terms of navigation steps) for 2 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 100 simulations.



(a) 1 step, $v_p = v_r$

(b) 3 steps, $v_p = v_r$

(c) 6 steps, $v_p = v_r$

(d) 1 step, $v_p = 2v_r$

(e) 3 steps, $v_p = 2v_r$

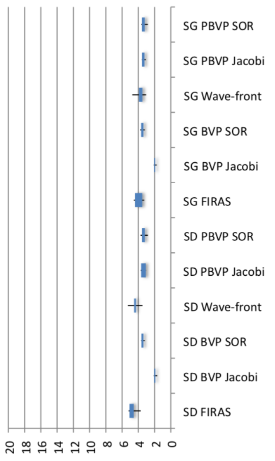
(f) 6 steps, $v_p = 2v_r$

(g) 1 step, $v_p = \frac{1}{2}v_r$

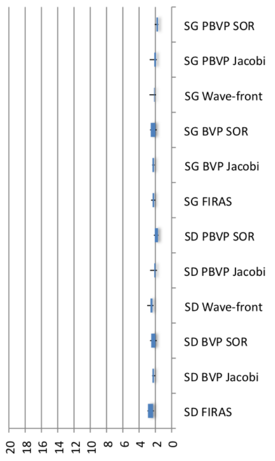
(h) 3 steps, $v_p = \frac{1}{2}v_r$

(i) 6 steps, $v_p = \frac{1}{2}v_r$

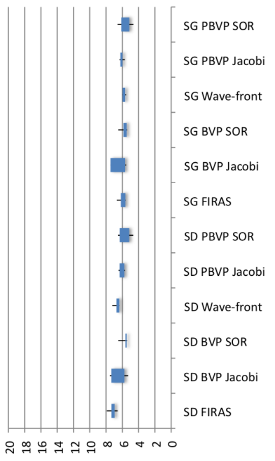
Figure E.8: Capture times (in terms of navigation steps) for 4 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 100 simulations.



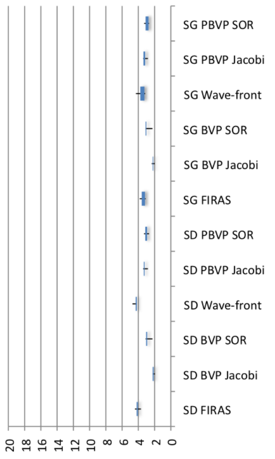
(c) 6 steps, $v_p = v_r$



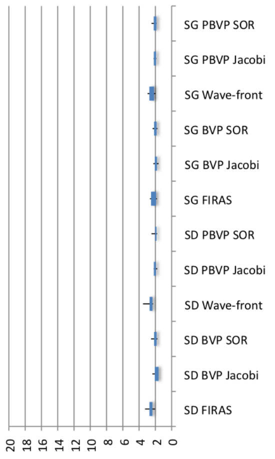
(f) 6 steps, $v_p = 2v_r$



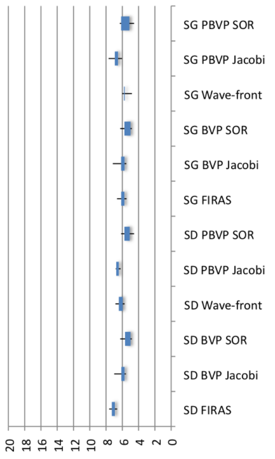
(i) 6 steps, $v_p = \frac{1}{2}v_r$



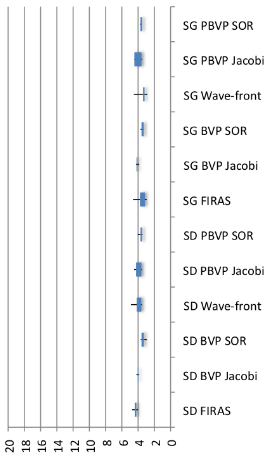
(b) 3 steps, $v_p = v_r$



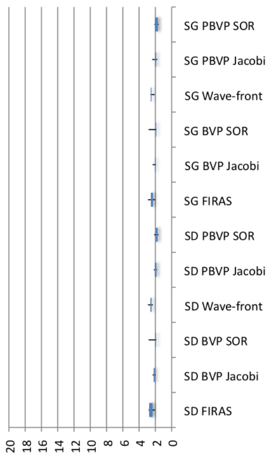
(e) 3 steps, $v_p = 2v_r$



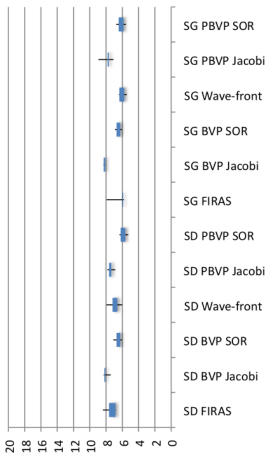
(h) 3 steps, $v_p = \frac{1}{2}v_r$



(a) 1 step, $v_p = v_r$

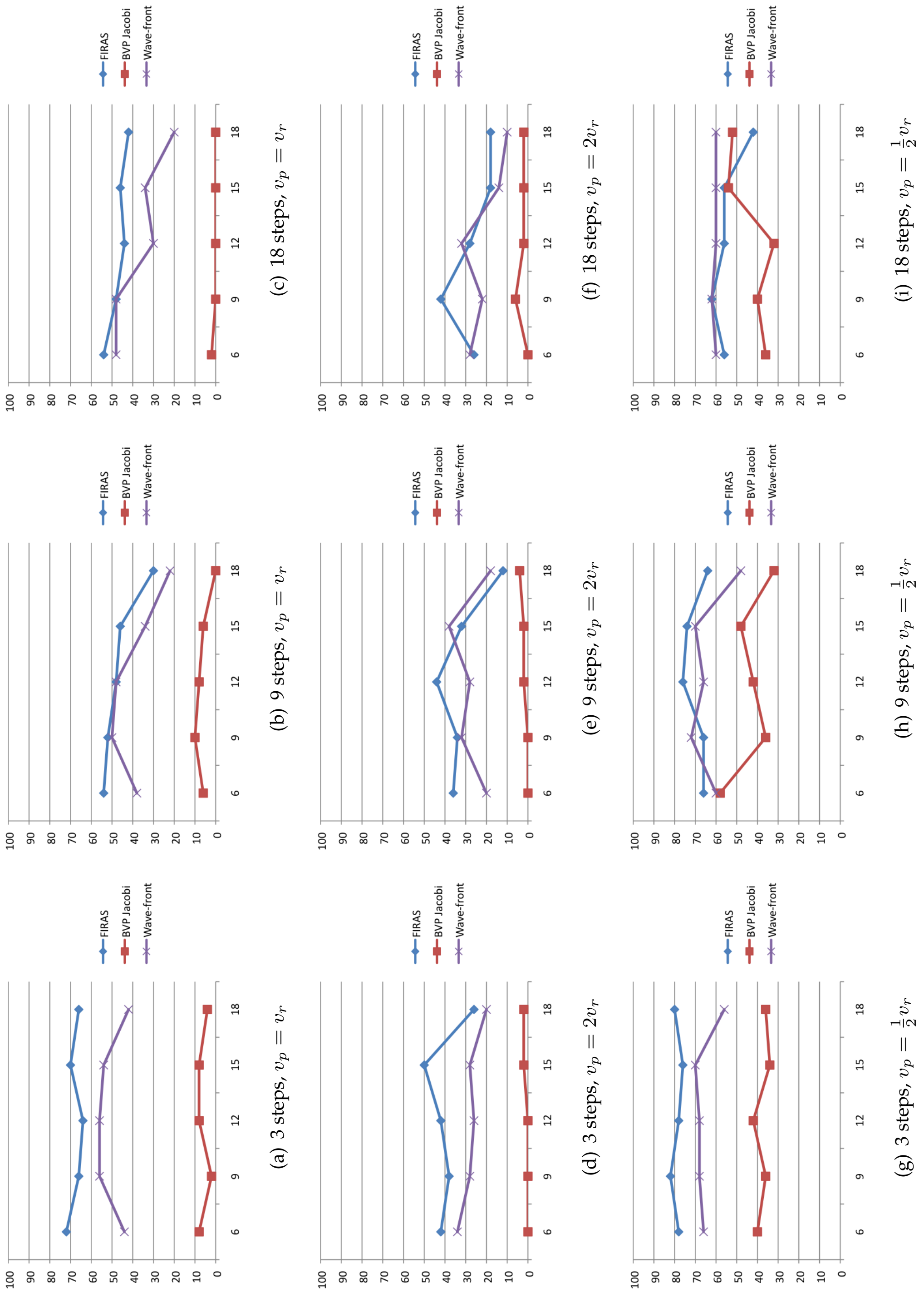


(d) 1 step, $v_p = 2v_r$



(g) 1 step, $v_p = \frac{1}{2}v_r$

Figure E.9: Capture times (in terms of navigation steps) for 5 pursuers, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 100 simulations.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

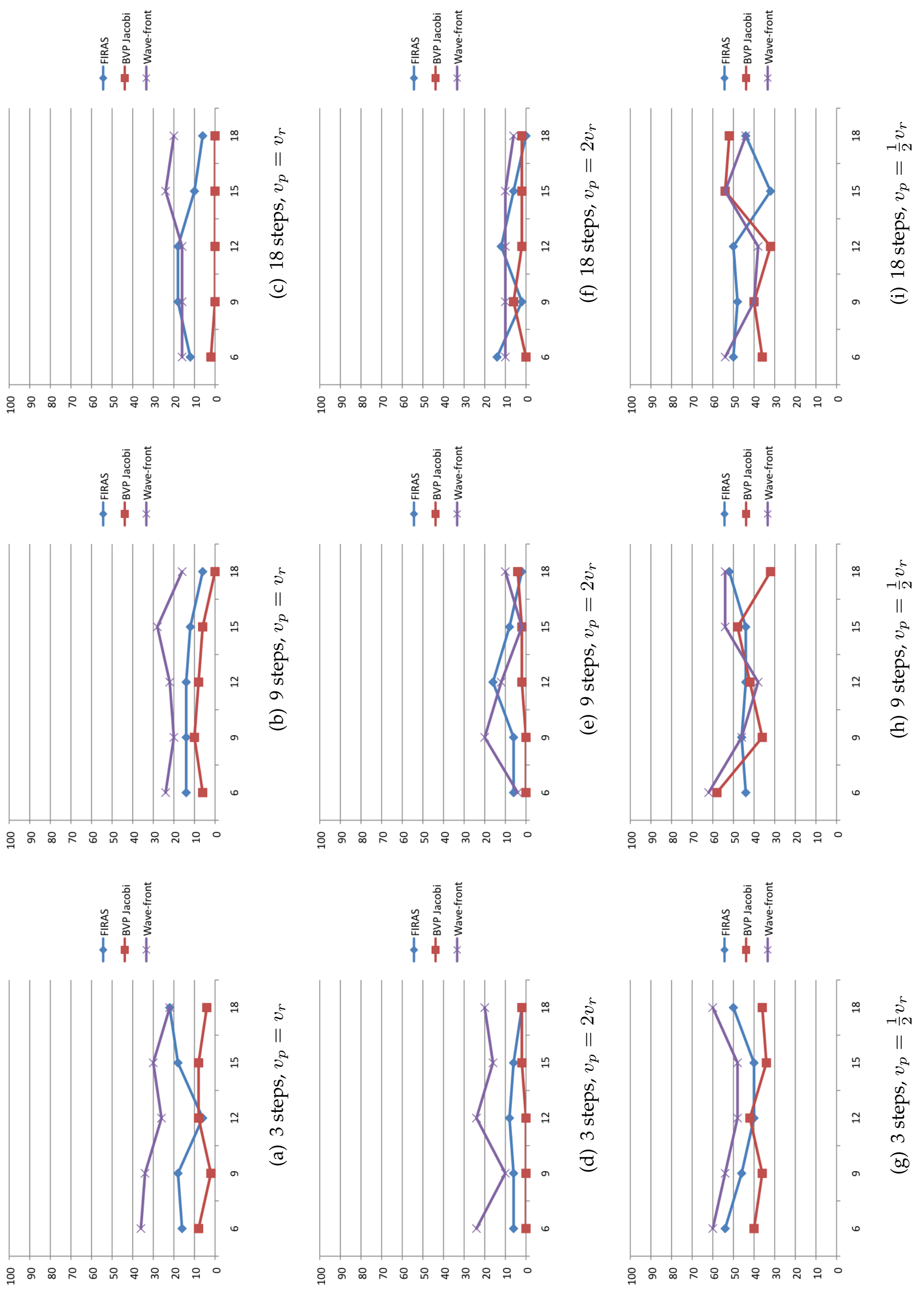
(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure E.10: Success of steepest descent paths for 2 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

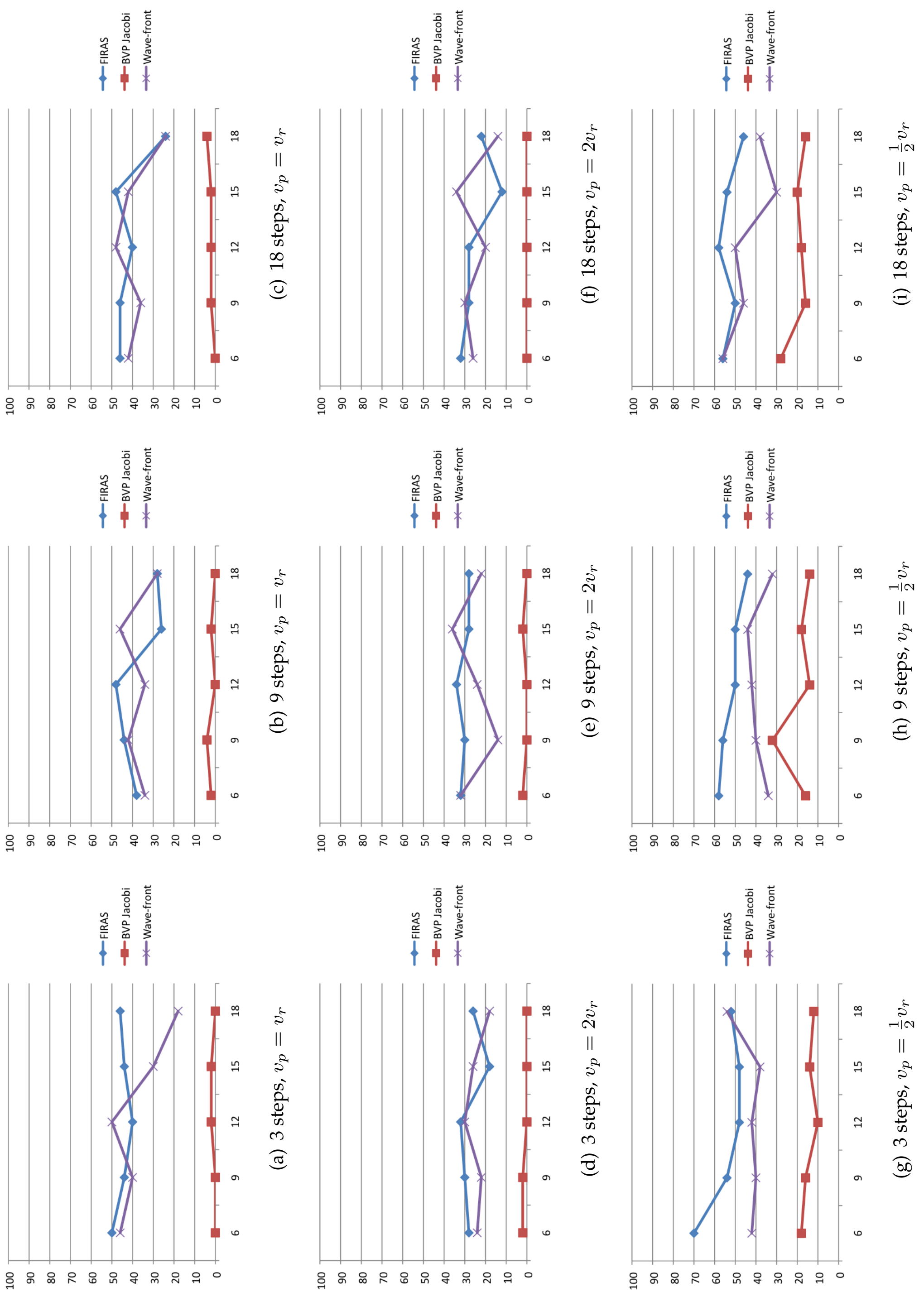
(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure E.11: Success of subgoal based paths for 2 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

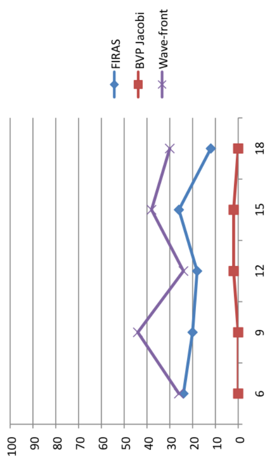
(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

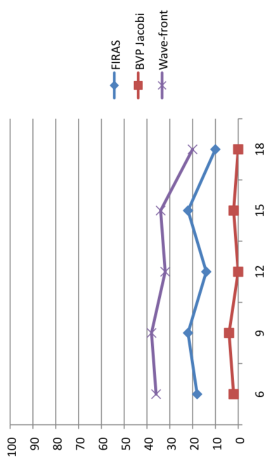
(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

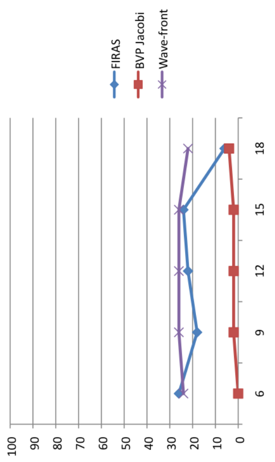
Figure E.12: Success of steepest descent paths for 4 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



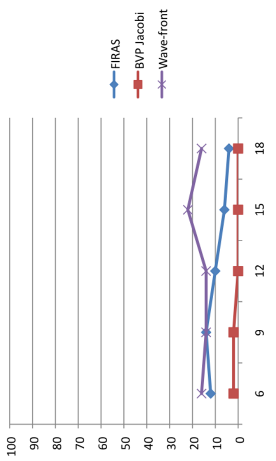
(a) 3 steps, $v_p = v_r$



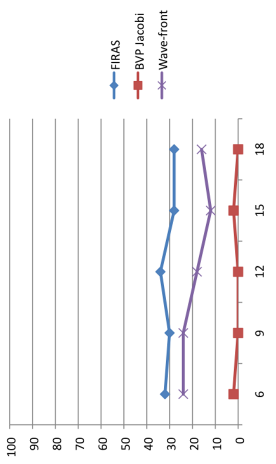
(b) 9 steps, $v_p = v_r$



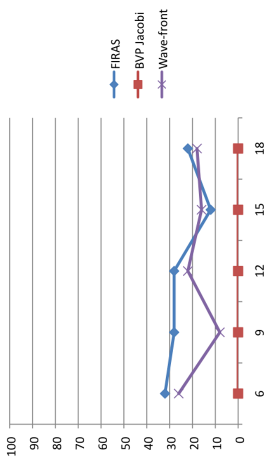
(c) 18 steps, $v_p = v_r$



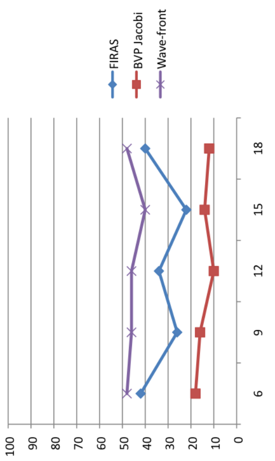
(d) 3 steps, $v_p = 2v_r$



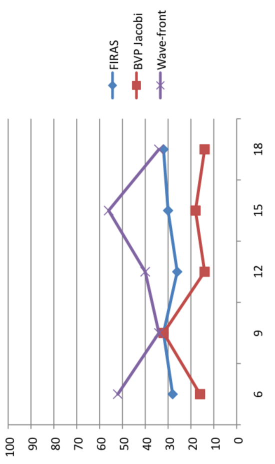
(e) 9 steps, $v_p = 2v_r$



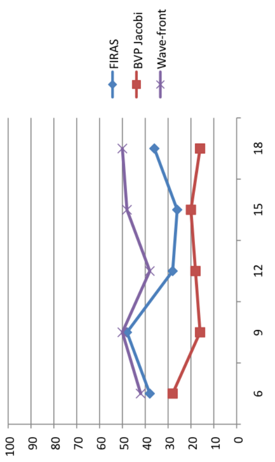
(f) 18 steps, $v_p = 2v_r$



(g) 3 steps, $v_p = \frac{1}{2}v_r$

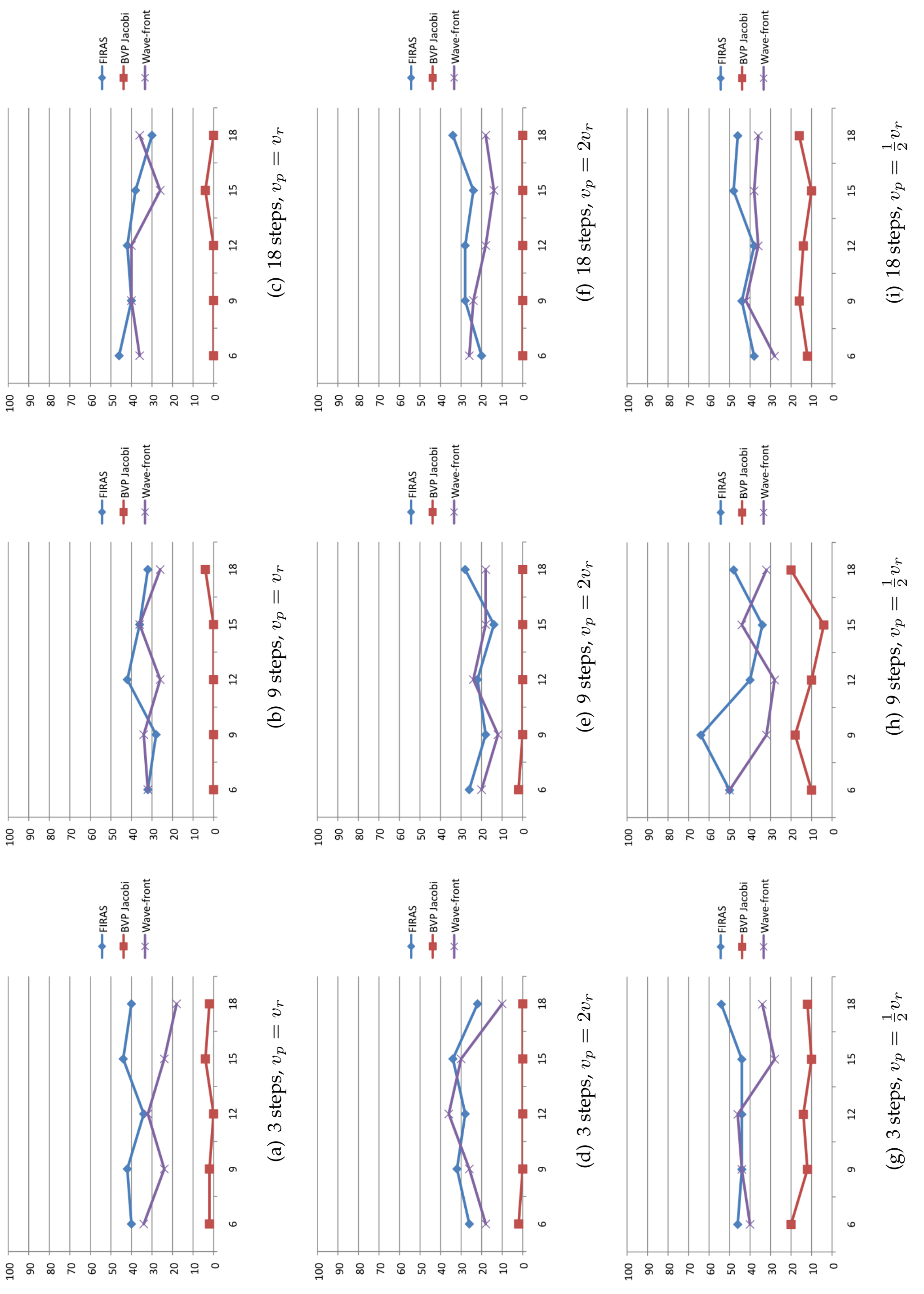


(h) 9 steps, $v_p = \frac{1}{2}v_r$



(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure E.13: Success of subgoal based paths for 4 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

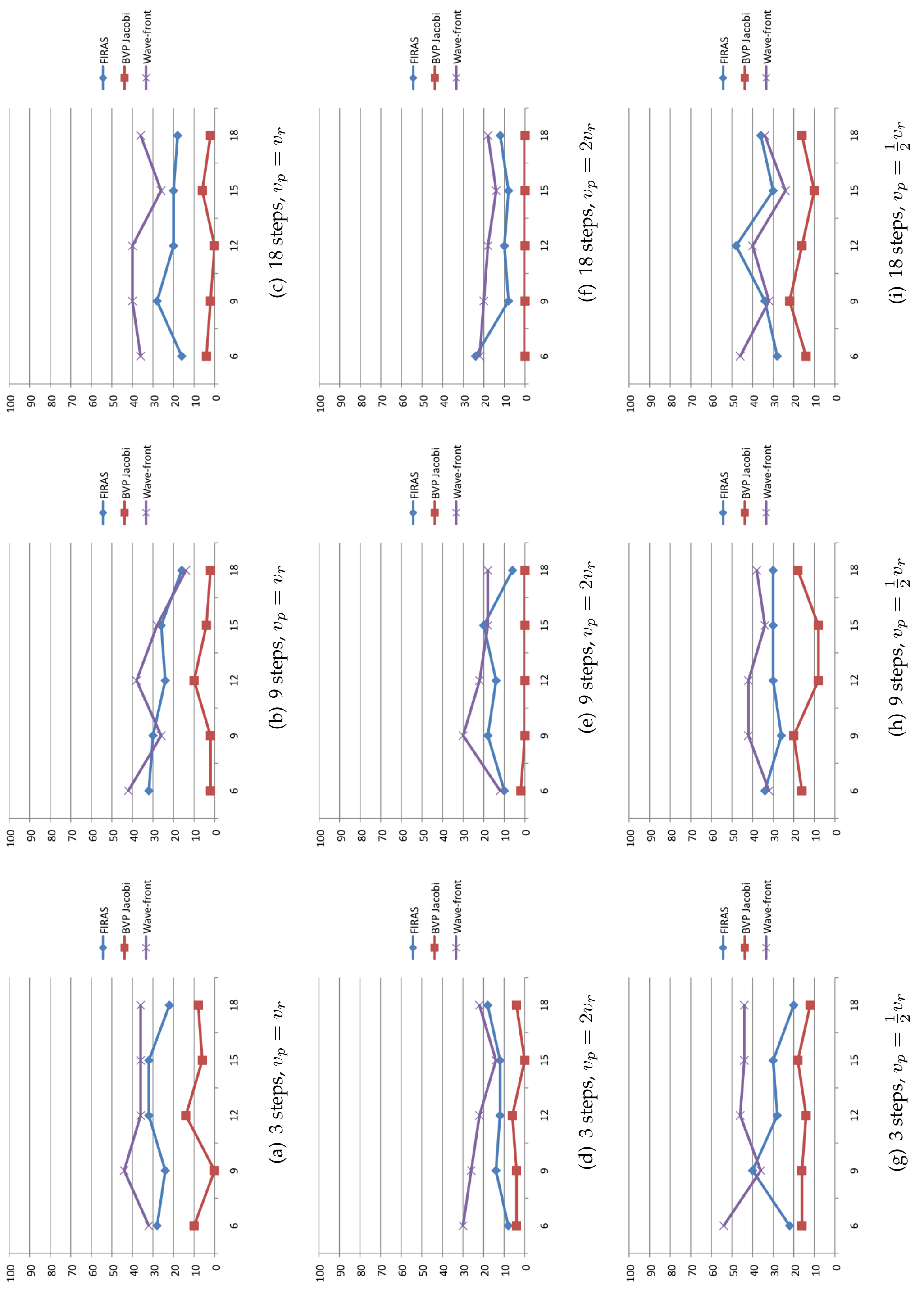
(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure E.14: Success of steepest descent paths for 5 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds.The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

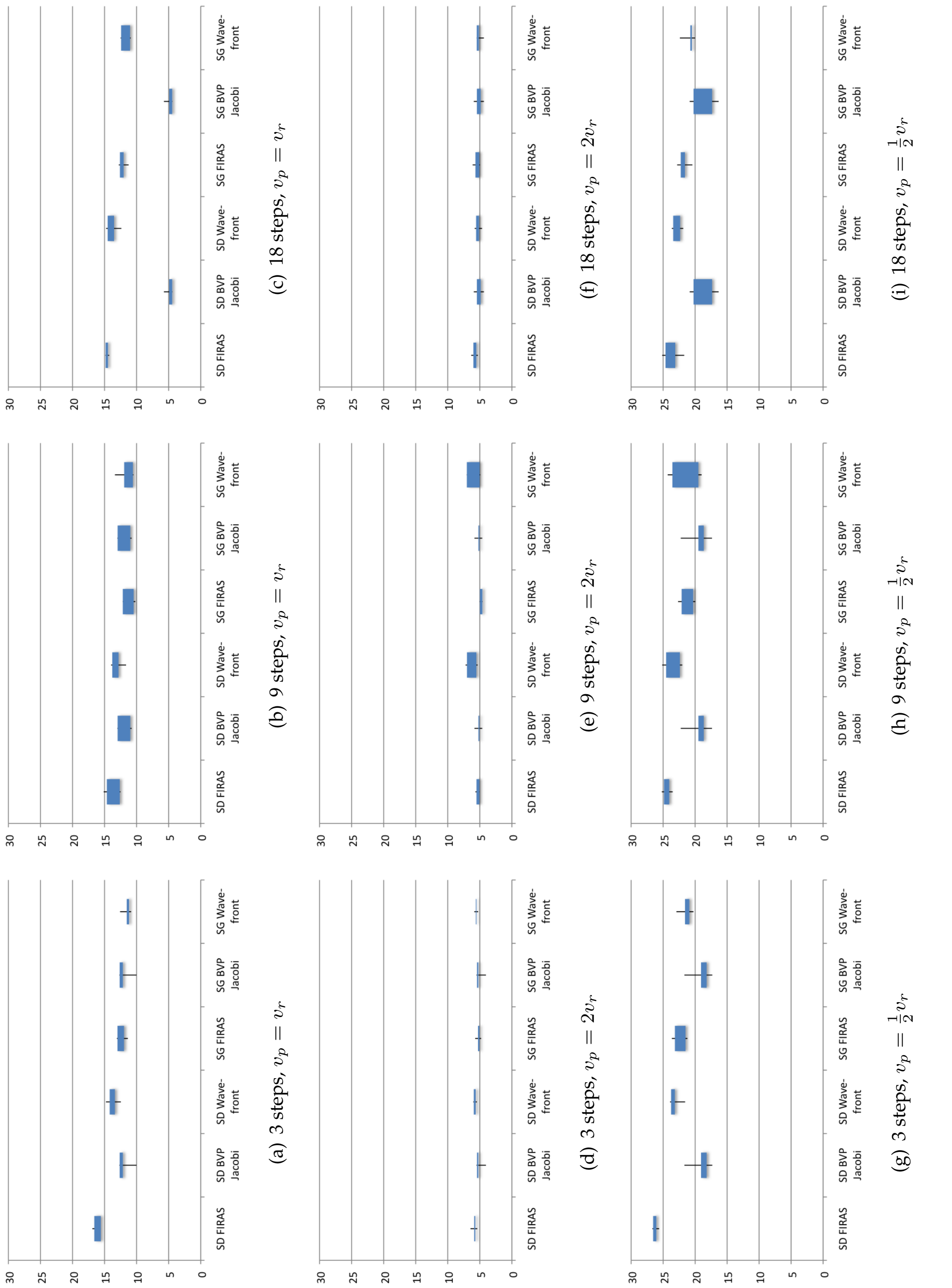
(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure E.15: Success of subgoal based paths for 5 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds.The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

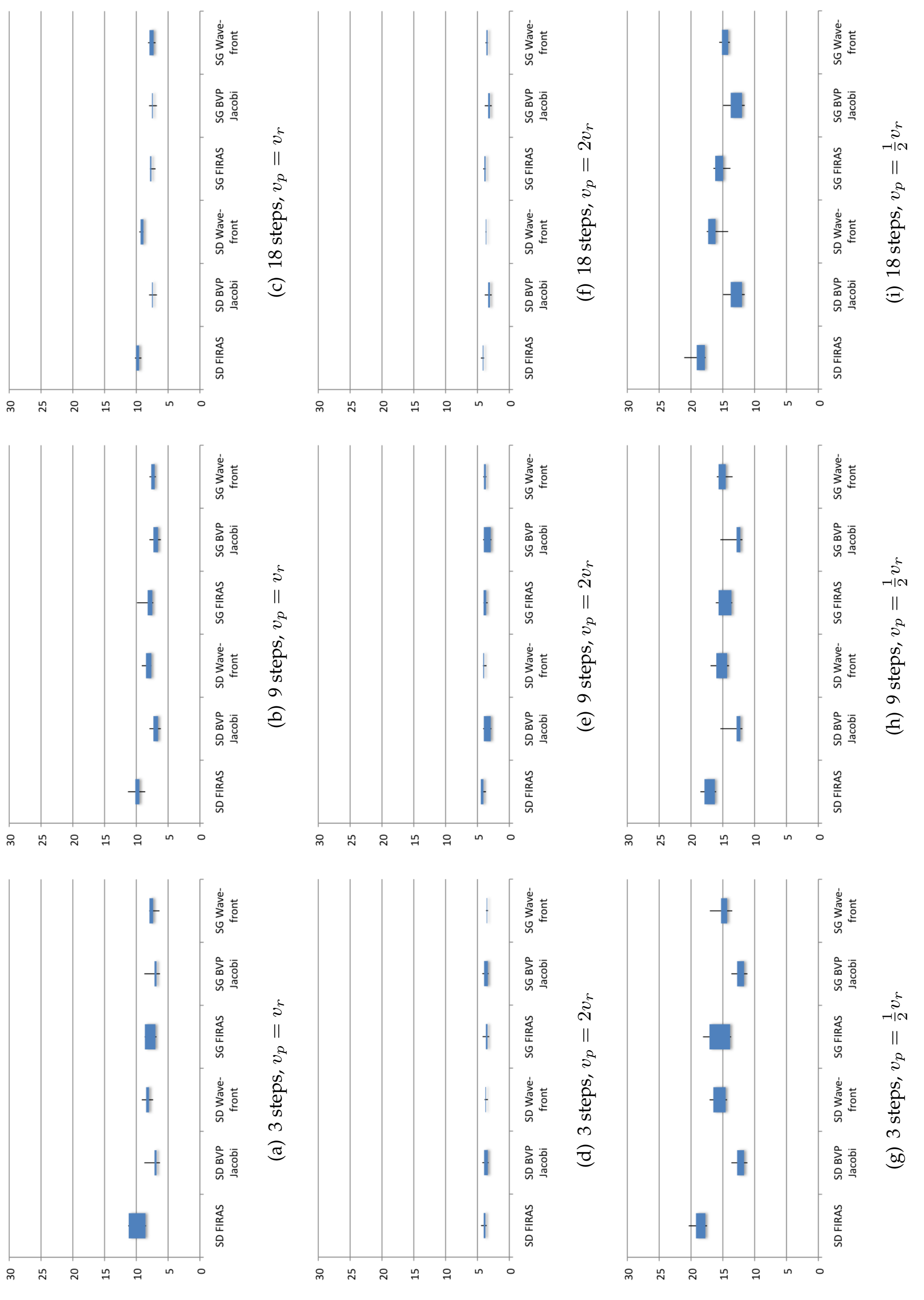
(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure E.16: Capture times (in terms of navigation steps) for 2 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 50 simulations.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure E.17: Capture times (in terms of navigation steps) for 4 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 50 simulations.

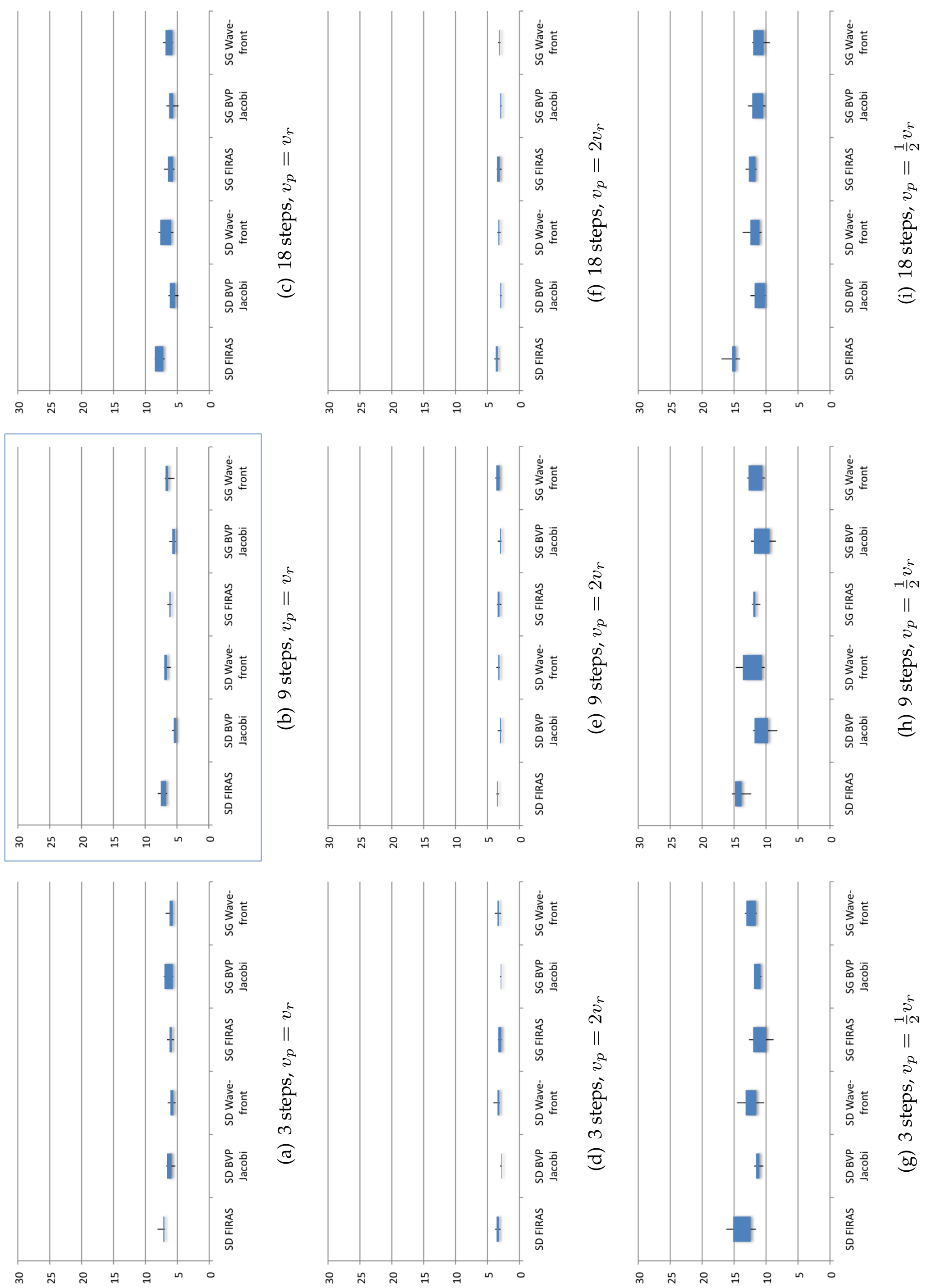
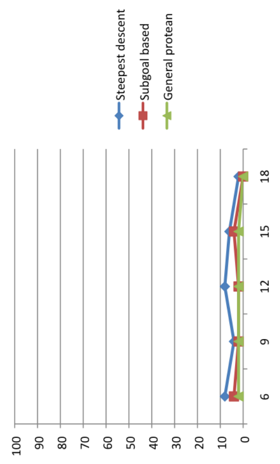
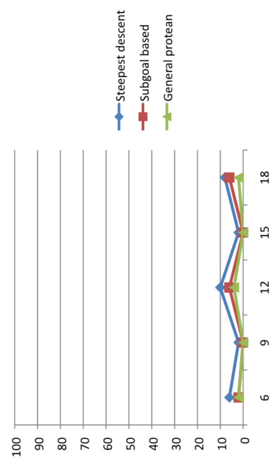


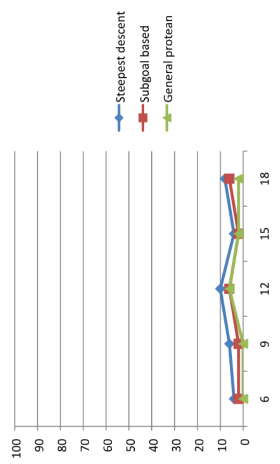
Figure E.18: Capture times (in terms of navigation steps) for 5 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 50 simulations.



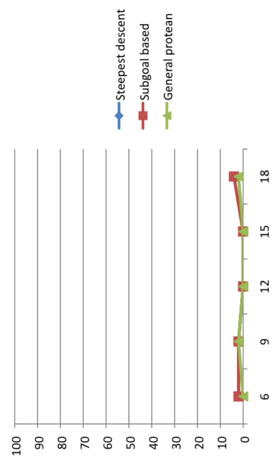
(a) 3 steps, $v_p = v_r$



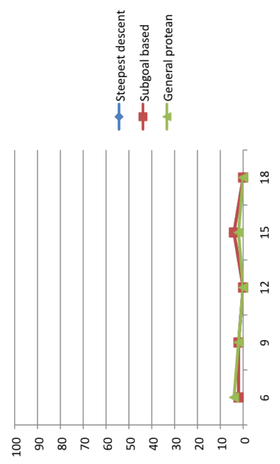
(b) 9 steps, $v_p = v_r$



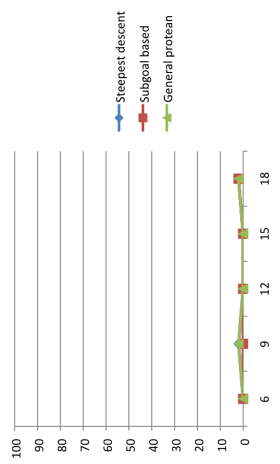
(c) 18 steps, $v_p = v_r$



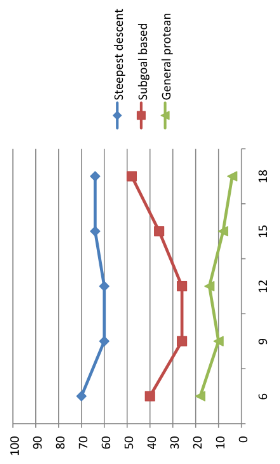
(d) 3 steps, $v_p = 2v_r$



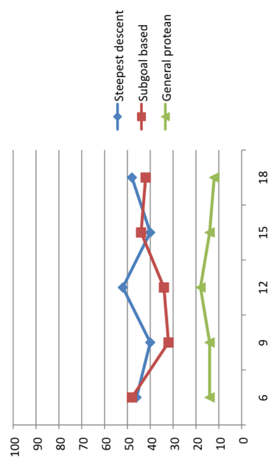
(e) 9 steps, $v_p = 2v_r$



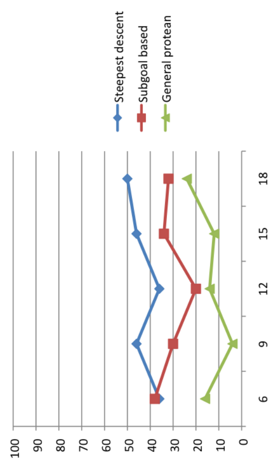
(f) 18 steps, $v_p = 2v_r$



(g) 3 steps, $v_p = \frac{1}{2}v_r$

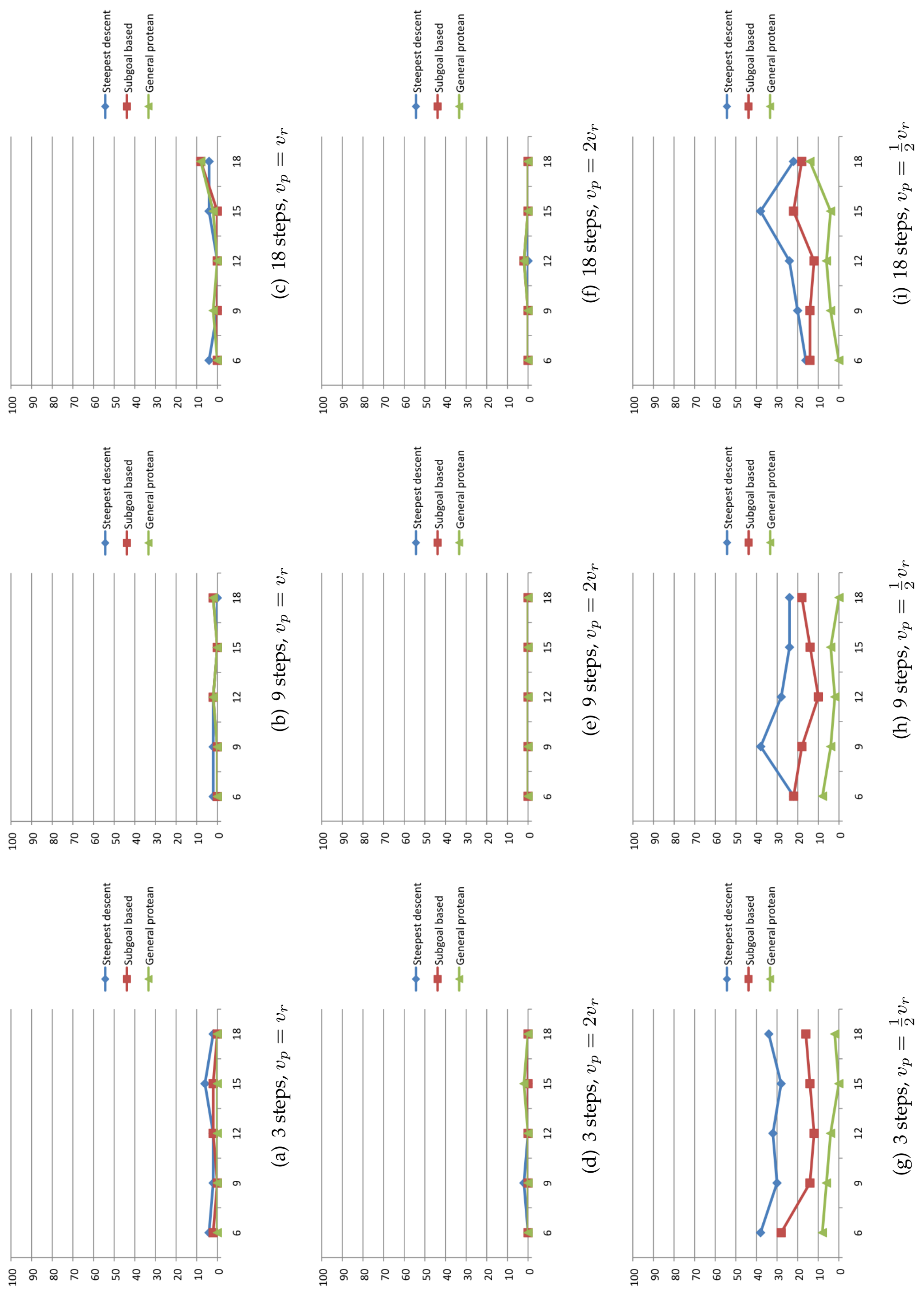


(h) 9 steps, $v_p = \frac{1}{2}v_r$



(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure E.19: Success of the paths for 2 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

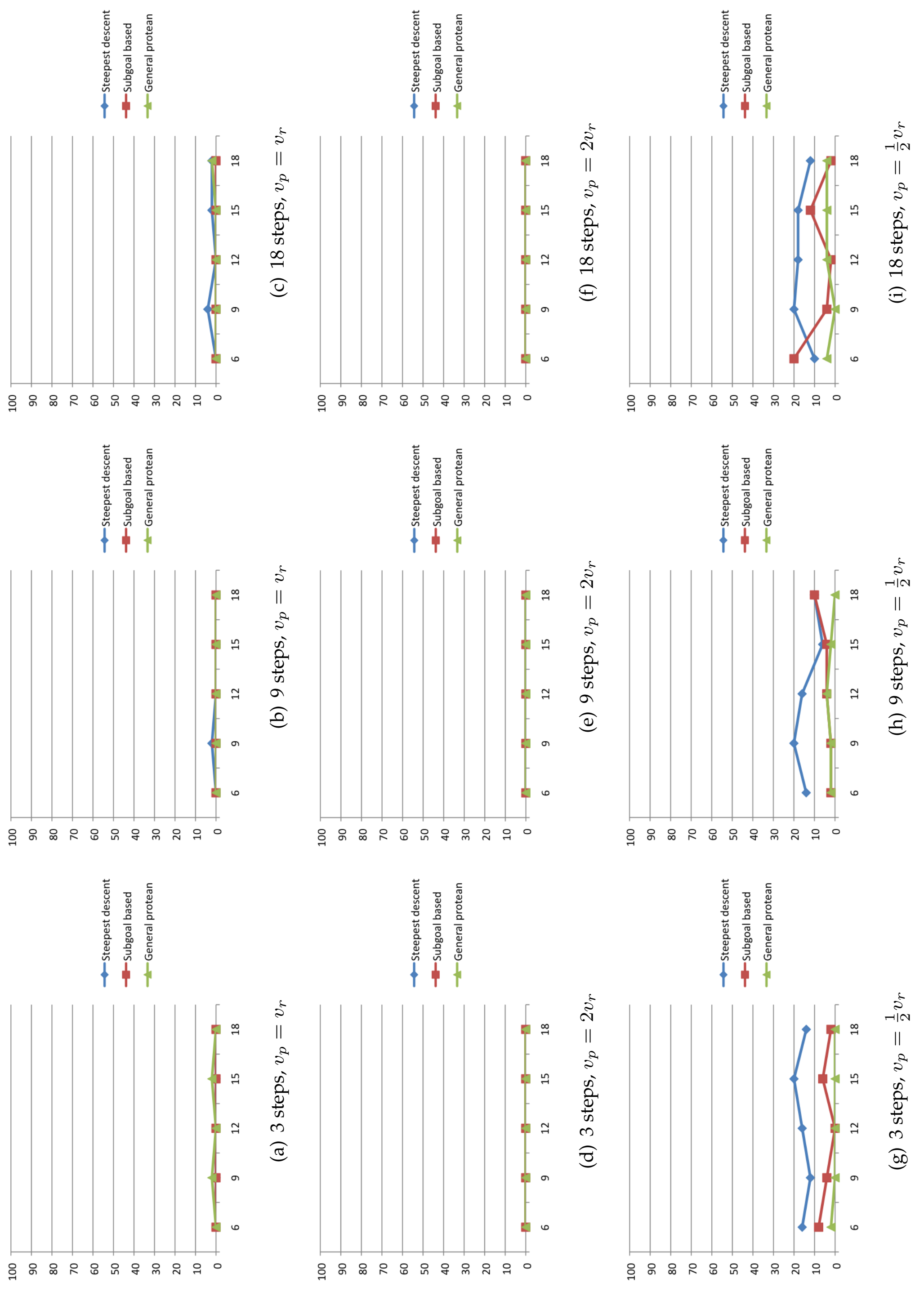
(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure E.20: Success of the paths for 4 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values of r_{sg} and the vertical axis is the percentage of success.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

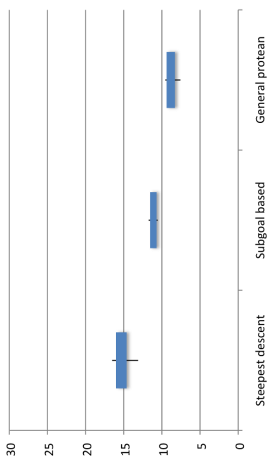
(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

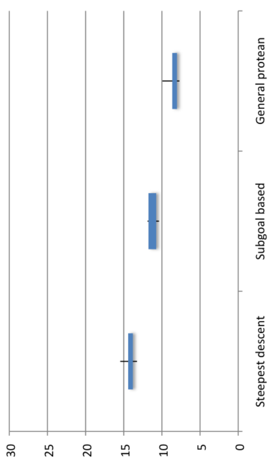
(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

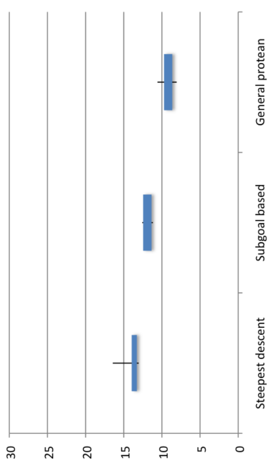
Figure E.21: Success of the paths for 5 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the values or r_{sg} and the vertical axis is the percentage of success.



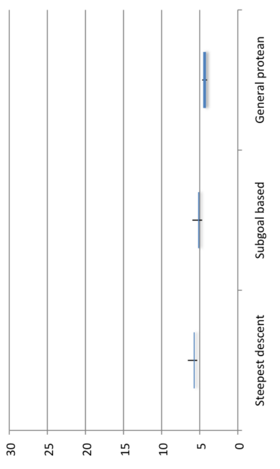
(a) 3 steps, $v_p = v_r$



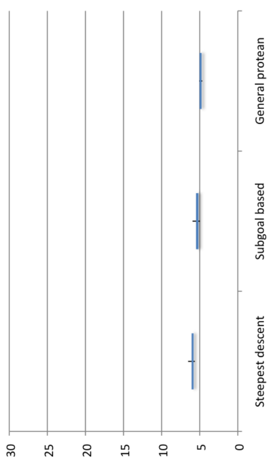
(b) 9 steps, $v_p = v_r$



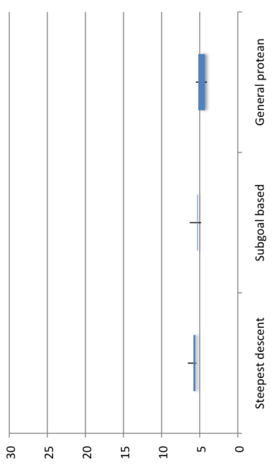
(c) 18 steps, $v_p = v_r$



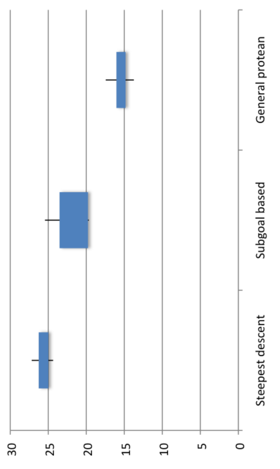
(d) 3 steps, $v_p = 2v_r$



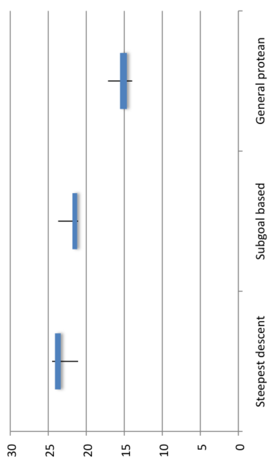
(e) 9 steps, $v_p = 2v_r$



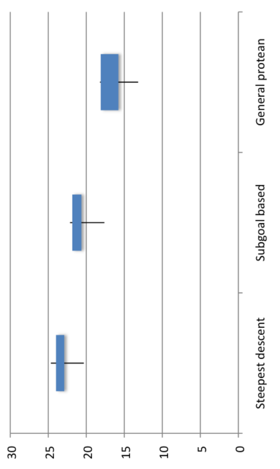
(f) 18 steps, $v_p = 2v_r$



(g) 3 steps, $v_p = \frac{1}{2}v_r$

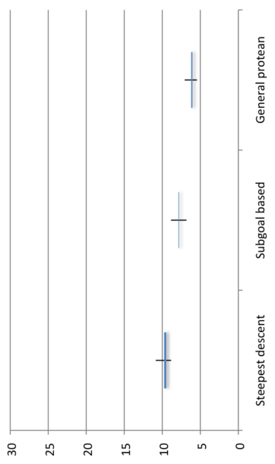


(h) 9 steps, $v_p = \frac{1}{2}v_r$

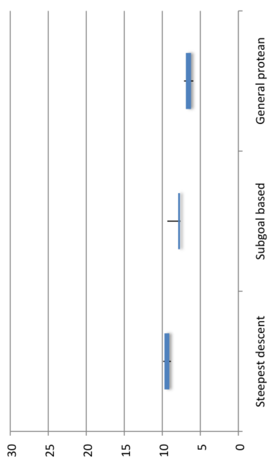


(i) 18 steps, $v_p = \frac{1}{2}v_r$

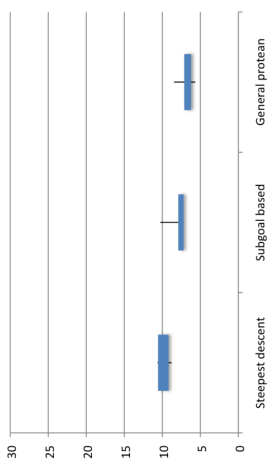
Figure E.22: Capture times (in terms of navigation steps) for 2 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 50 simulations.



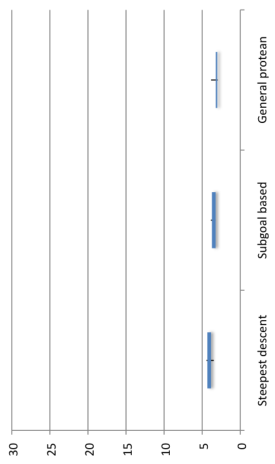
(a) 3 steps, $v_p = v_r$



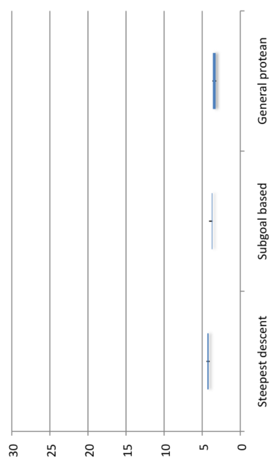
(b) 9 steps, $v_p = v_r$



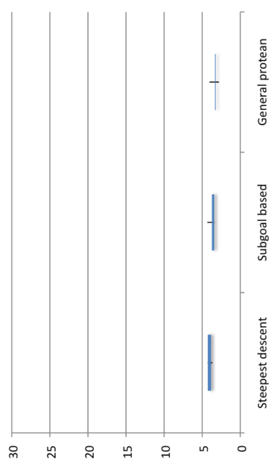
(c) 18 steps, $v_p = v_r$



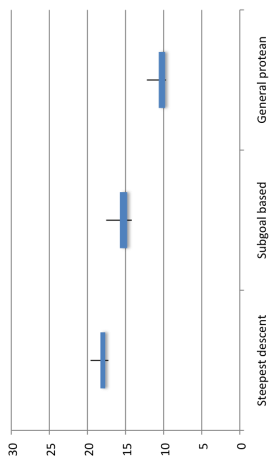
(d) 3 steps, $v_p = 2v_r$



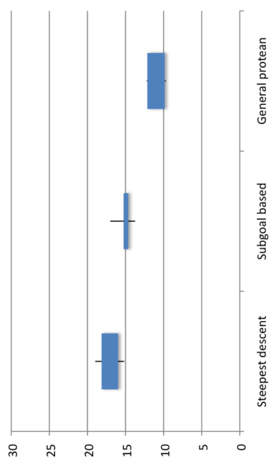
(e) 9 steps, $v_p = 2v_r$



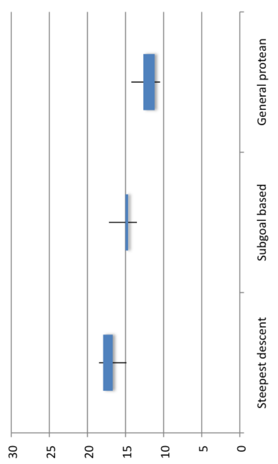
(f) 18 steps, $v_p = 2v_r$



(g) 3 steps, $v_p = \frac{1}{2}v_r$

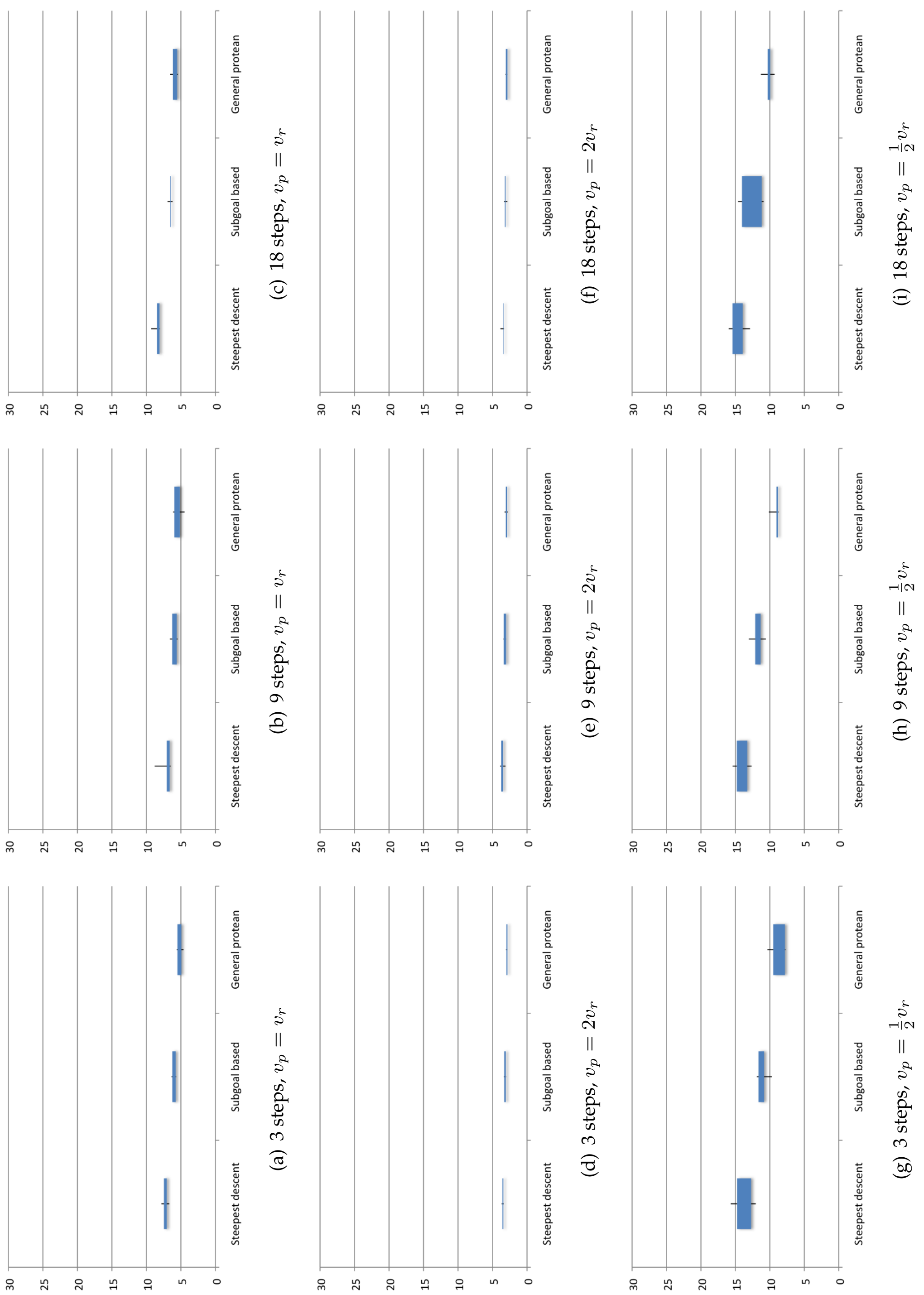


(h) 9 steps, $v_p = \frac{1}{2}v_r$



(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure E.23: Capture times (in terms of navigation steps) for 4 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 50 simulations.



(a) 3 steps, $v_p = v_r$

(b) 9 steps, $v_p = v_r$

(c) 18 steps, $v_p = v_r$

(d) 3 steps, $v_p = 2v_r$

(e) 9 steps, $v_p = 2v_r$

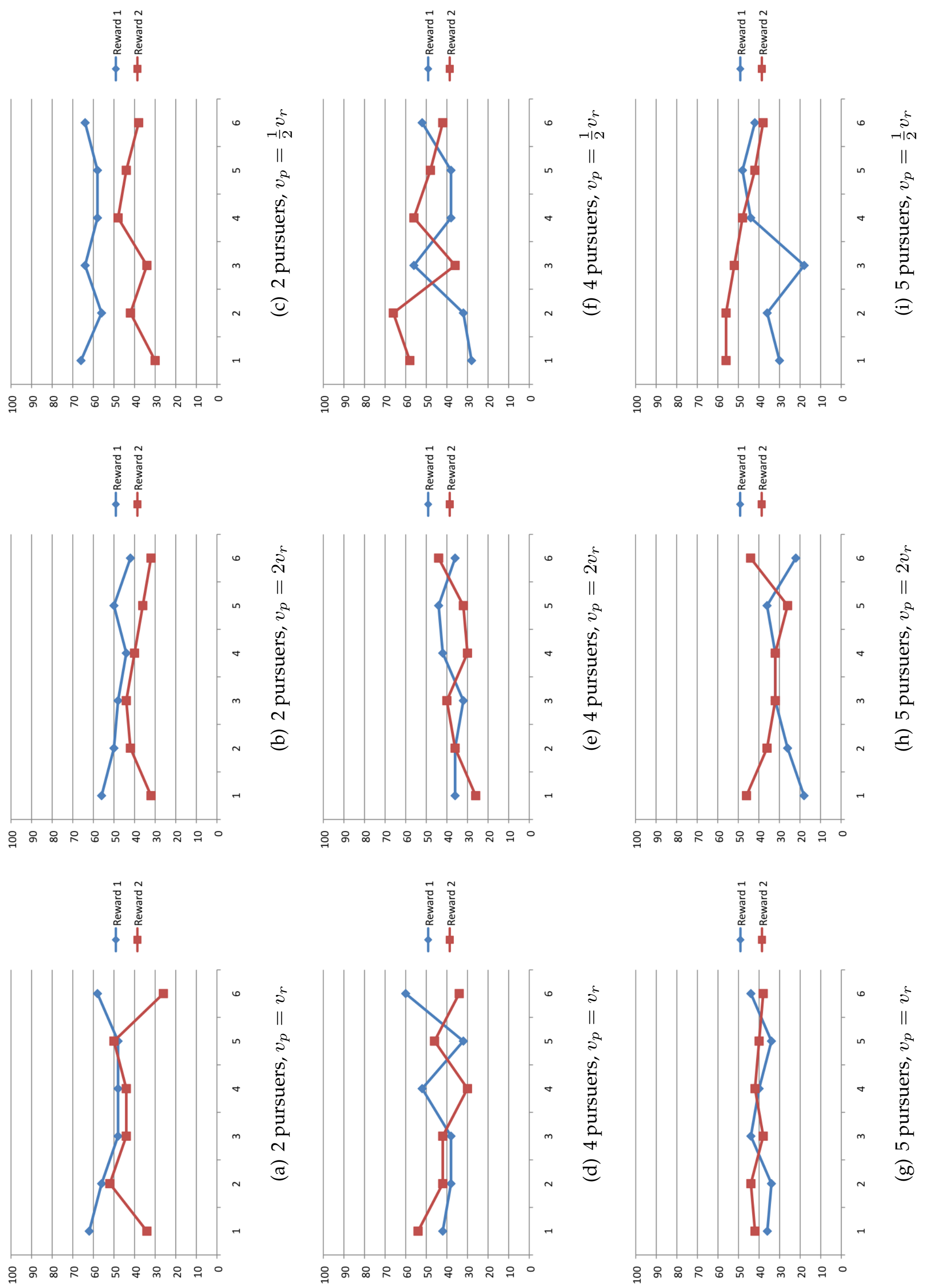
(f) 18 steps, $v_p = 2v_r$

(g) 3 steps, $v_p = \frac{1}{2}v_r$

(h) 9 steps, $v_p = \frac{1}{2}v_r$

(i) 18 steps, $v_p = \frac{1}{2}v_r$

Figure E.24: Capture times (in terms of navigation steps) for 5 pursuers, a higher resolution grid, different sensing updates (measured in navigation steps) and speeds. The horizontal axis corresponds to the used potential functions, and the vertical axis is the average of navigation steps before capture in 50 simulations.



(a) 2 pursuers, $v_p = v_r$

(b) 2 pursuers, $v_p = 2v_r$

(c) 2 pursuers, $v_p = \frac{1}{2}v_r$

(d) 4 pursuers, $v_p = v_r$

(e) 4 pursuers, $v_p = 2v_r$

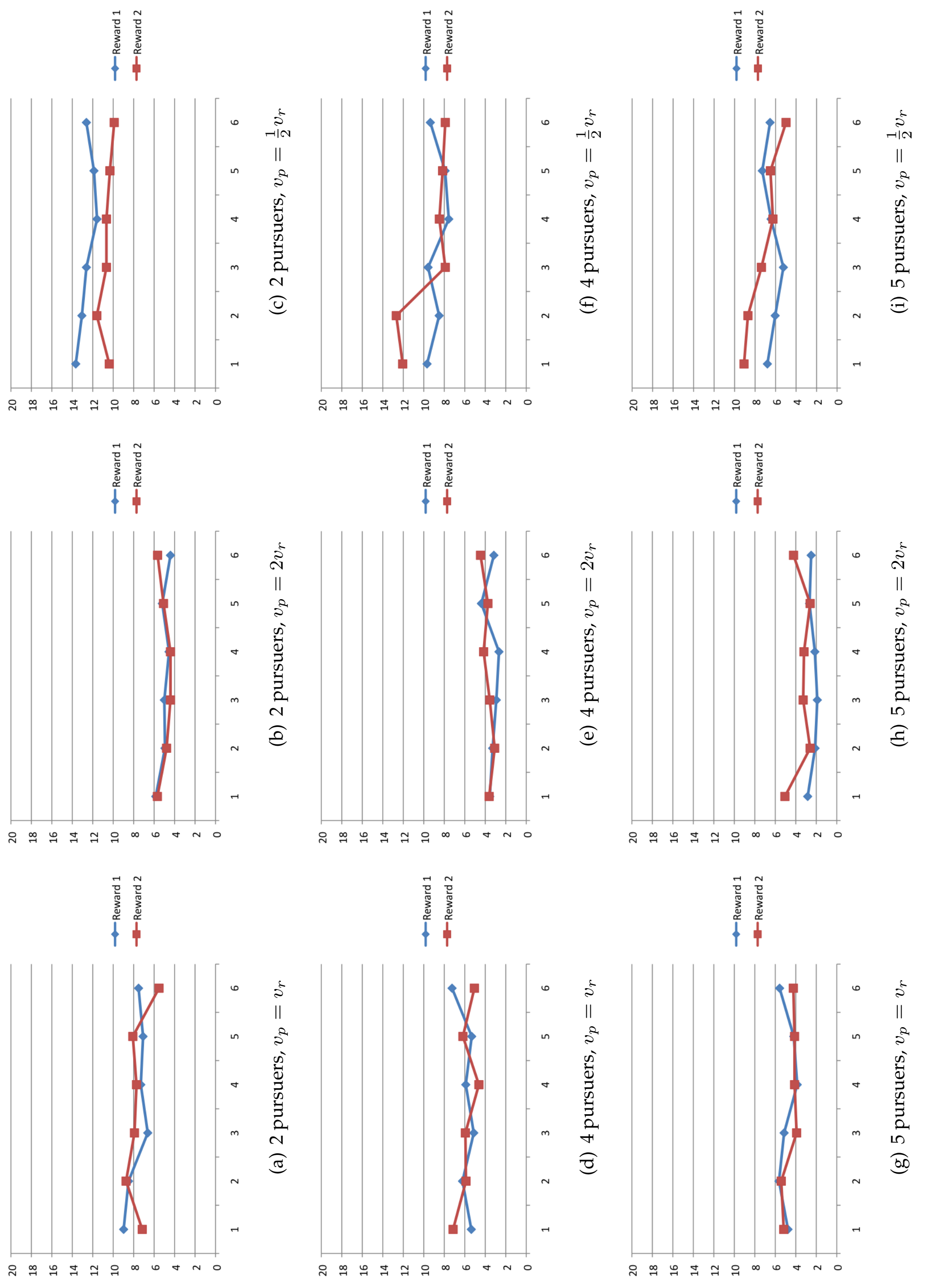
(f) 4 pursuers, $v_p = \frac{1}{2}v_r$

(g) 5 pursuers, $v_p = v_r$

(h) 5 pursuers, $v_p = 2v_r$

(i) 5 pursuers, $v_p = \frac{1}{2}v_r$

Figure E.25: Success of the strategies for 2, 4 and 5 pursuers with different velocities. The horizontal axis corresponds to the sensing updates (in navigation steps) and the vertical axis is the percentage of success.



(a) 2 pursuers, $v_p = v_r$

(b) 2 pursuers, $v_p = 2v_r$

(c) 2 pursuers, $v_p = \frac{1}{2}v_r$

(d) 4 pursuers, $v_p = v_r$

(e) 4 pursuers, $v_p = 2v_r$

(f) 4 pursuers, $v_p = \frac{1}{2}v_r$

(g) 5 pursuers, $v_p = v_r$

(h) 5 pursuers, $v_p = 2v_r$

(i) 5 pursuers, $v_p = \frac{1}{2}v_r$

Figure E.26: Capture times (in terms of navigation steps), for 2, 4 and 5 pursuers with different velocities. The horizontal axis corresponds to the sensing updates (in navigation steps) and the vertical axis is the average of capture time (also in navigation steps), for 50 simulations.

Bibliography

P.A. Abrams. The evolution of predator-prey interactions: theory and evidence. *Annual Review on Ecological Systems*, 31(1):79–105, 2000.

T.G. Abramyants, M.N. Ivanov, E.P. Maslov, and V.P. Yakhno. A detection evasion problem. *Automation and Remote Control*, 65(10):1523–1530, 2004.

M. Adler, H. Räcke, N. Sivadasan, C. Sohler, and B. Vöcking. Randomized pursuit-evasion in graphs. *Combinatorics, Probability and Computing*, 12(3):225–244, 2003.

M.R. Akbarzadeh, H. Rezaei, and M.B. Naghibi. A fuzzy adaptive algorithm for expertness based cooperative learning, application to herding problem. In *Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society*, pages 317–322, Chicago, Illinois, USA, 2003.

K.S. Al-Sultan and M.D.S. Aliyu. A new potential field-pased algorithm for path planning. *Journal of Intelligent and Robotic Systems*, 17(3):265–282, 1996.

A. Altahhan, K. Burn, and S. Wermter. Visual robot homing using Sarsa(λ), whole image measure, and radial basis function. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 3861–3868, Hong Kong, China, 2008.

D. Alvarez, J.C. Alvarez, and R.C. González. Online motion planning using Laplace potential fields. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3347–3352, Taipei, Taiwan, 2003.

S.M. Amin, E.Y. Rodin, M.K. Meusey, T.W. Cusick, and A. García-Ortiz. Evasive adaptive navigation and control against multiple pursuers. In *Proceedings of the American Control Conference*, pages 1453–1457, Albuquerque, New Mexico, USA, 1997.

A.J. Anderson and P.W. McOwan. Model of a predatory stealth behaviour camouflaging motion. *Proceedings of the Royal Society B: Biological Sciences*, 270(1514):489–495, 2003.

A. Antoniades, H.J. Kim, and S. Sastry. Pursuit-evasion strategies for teams of multiple agents with incomplete information. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 756–761, Maui, Hawaii, USA, 2003.

D. Araiza-Illan and T.J. Dodd. Biologically inspired controller for the autonomous navigation of a mobile robot in an evasion task. *World Academy of Science, Engineering and Technology*, 44:780–785, 2010.

R. Armour, K. Paskins, A. Bowyer, J. Vincent, and W. Megill. Jumping robots: a biomimetic solution to locomotion across rough terrain. *Bioinspiration and Biomimetics*, 2(3):65–82, 2007.

M. Asada, E. Uchibe, and K. Hosoda. Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development. *Artificial Intelligence*, 110(2):275–292, 1999.

R.J. Bachmann, F.J. Boria, R. Vaidyanathan, P.G. Ifju, and R.D. Quinn. A biologically inspired micro-vehicle capable of aerial and terrestrial locomotion. *Mechanism and Machine Theory*, 44(3):513–526, 2009.

T. Başar and G.J. Olsder. *Dynamic noncooperative game theory*. Society for Industrial and Applied Mathematics, 1999.

Y. Bar-Cohen and C. Breazeal. Biologically inspired intelligent robotics. In *Proceedings of the SPIE Smart Structures Conference*, pages 1–7, San Diego, California, USA, 2003.

V. Baranauskas, K. Sarkauskas, and S. Bartkevicius. Creation of vector marks for robot navigation. *Elektronika ir Elektrotechnika*, 4(84):27–30, 2008.

J. Barraquand, B. Langlois, and J.C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(2):224–241, 1992.

A. Barrera-Ramirez and A. Weitzenfeld-Ridel. Biologically-inspired robot spatial cognition based on rat neurophysiological studies. *Autonomous Robots*, 25(1-2):147–169, 2008.

A.G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, 13(1-2):343–379, 2003.

R.D. Beer, R.D. Quinn, H.J. Chiel, and R.E. Ritzmann. Biologically inspired approaches to robotics. *Communications of the Association for Computing Machinery*, 40(3):32–38, 1997.

G.A. Bekey. Biologically inspired control of autonomous robots. *Robotics and Autonomous Systems*, 18(1-2):21–31, 1996.

G. Bell. *Forward chaining for potential field based navigation*. PhD thesis, University of St Andrews, 2005.

H.R. Berenji. Fuzzy Q-learning: a new approach for fuzzy dynamic programming. In *Proceedings of the 3rd IEEE Conference on Fuzzy Systems*, pages 486–491, Orlando, Florida, USA, 1994.

P. Bernhard and O. Pourtallier. Pursuit evasion game with costly information. *Dynamics and Control*, 4(4):365–382, 1994.

K. Berns, W. Ilg, M. Deck, J. Albiez, and R. Dillmann. Mechanical construction and computer architecture of the four-legged walking machine BISAM. *IEEE/ASME Transactions on Mechatronics*, 4(1):32–38, 1999.

A.A. Berryman. The origins and evolution of predator-prey theory. *Ecology*, 73(5):1530–1535, 1992.

S. Bhattacharya, S. Hutchinson, and T. Başar. Game-theoretic analysis of a visibility based pursuit-evasion game in the presence of obstacles. In *Proceedings of the American Control Conference*, pages 373–378, St. Louis, Missouri, USA, 2009.

E. Birgersson, A. Howard, and G.S. Sukhatme. Towards stealthy behaviors. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1703–1708, Las Vegas, Nevada, USA, 2003.

S.D. Bopardikar, F. Bullo, and J. Hespanha. Cooperative pursuit with sensing limitations. In *Proceedings of the American Control Conference*, pages 5394–5399, New York, New York, USA, 2007.

S.D. Bopardikar, F. Bullo, and J.P. Hespanha. A coopeartive homicidal chauffeur game. *Automatica*, 45(7):1771–1777, 2009.

J. Borenstein and Y. Koren. The vector field histogram- fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, 1991.

G. Borghesan, G. Palli, and C. Melchiorri. Design of tendon-driven robotic fingers: Modeling and control issues. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 793–798, Anchorage, Alaska, USA, 2010.

A.T. Bradshaw and L.L. Moseley. Dynamics of competing predator-prey species. *Physica A*, 261(1-2):107–114, 1998.

P. Brass, K.D. Kim, H. Na, and C. Shin. Escaping offline searchers and isoperimetric theorems. *Computational Geometry: Theory and Applications*, 42(2):119–126, 2009.

M. Broom and G.D. Ruxton. You can run—or you can hide: optimal strategies for cryptic prey against pursuit predators. *Behavioural Ecology*, 16(3):534–540, 2005.

J.S. Brown, J.W. Laundré, and M. Gurung. The ecology of fear: Optimal foraging, game theory, and trophic interactions. *Journal of Mammalogy*, 80(2):385–399, 1999.

J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2383–2388, Lausanne, Switzerland, 2002.

R. Buckdahn, P. Cardaliaguet, and M. Quincampoix. Some recent aspects of differential game theory. *Dynamic Games and Applications*, 1(1):74–114, 2011.

L. Buşoniu, R. Babuška, and B.D. Schutter. Multi-agent reinforcement learning: a survey. In *Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision*, pages 1–6, Singapore, Singapore, 2006.

L. Buşoniu, R. Babuška, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics- Part C: Applications and Reviews*, 38(2):156–172, 2008.

Z. Cai, L. Sun, H. Gao, and P. Zhou. Multi-robot cooperative pursuit based on combinatorial auction mechanism under dynamic environment. In *Proceedings of the 2nd International*

Symposium on Systems and Control in Aerospace and Astronautics, pages 106–111, Shenzhen, China, 2008.

Z. Cai, L. Sun, and H. Gao. A novel hierarchical decomposition for multi-player pursuit evasion differential game with superior evaders. In *Proceedings of the World Summit on Genetic and Evolutionary Computation*, pages 795–798, Shanghai, China, 2009.

Y.U. Cao, A.S. Fukunaga, and A.B. Kahng. Cooperative mobile robotics: antecedents and directions. *Autonomous Robots*, 4(1):1–23, 1997.

P. Cardaliaguet. Nonsmooth semipermeable barriers, Isaacs' equation, and application to a differential game with one target and two players. *Applied Mathematics and Optimization*, 36(2):125–146, 1997.

T. Caro. *Antipredator defenses in birds and mammals*. The University of Chicago Press, 2005.

S. Caselli, M. Reggiani, and R. Sbravati. Parallel path planning with multiple evasion strategies. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 260–266, Washington, D.C., USA, 2002.

Y. Cen, L. Wang, and Z. Handong. Real-time obstacle avoidance strategy for mobile robot based on improved coordinating potential field with genetic algorithm. In *Proceedings of the IEEE Conference on Control Applications*, pages 519–523, Singapore, Singapore, 2007.

H. Chang. A new technique to handle local minimum for imperfect potential field based motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 108–112, Minneapolis, Minnesota, USA, 1996.

S. Charifa and M. Bikdash. Comparison of geometrical, kinematic, and dynamic performance of several potential field methods. In *Proceedings of the IEEE Southeast Conference*, pages 18–23, Atlanta, Georgia, USA, 2009.

C. Chen, H. Li, and D. Dong. Hybrid control for robot navigation. *IEEE Robotics and Automation Magazine*, pages 37–47, 2008.

O.Y. Cherkasov and A.G. Yakushev. Singular arcs in the optimal evasion against a proportional navigation vehicle. *Journal of Optimization Theory and Applications*, 113(2): 211–226, 2002.

A. Cherubini and F. Chaumette. A redundancy-based approach for obstacle avoidance in mobile robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5700–5705, Taipei, Taiwan, 2010.

T.S. Chingtham and S.B. Nair. *Multi-agent systems for society*, chapter Modeling a multiagent mobile robotics test bed using a biologically inspired artificial immune system, pages 270–283. Springer Verlag Berlin / Heidelberg, 2009.

H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of robot motion. Theory, algorithms and implementations*. MIT Press, 2005.

C.F. Chung and T. Furukawa. Coordinated pursuer control using particle filters for autonomous search-and-capture. *Robotics and Autonomous Systems*, 57(6-7):700–711, 2009.

H. Chung and J. Liu. Adaptive learning approach of fuzzy logic controller with evolution for pursuit-evasion games. In *Proceedings of the 2nd International Conference on Computational Collective Intelligence*, pages 482–490, Gdynia, Poland, 2010.

T.H. Chung, G.A. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics: a survey. *Autonomous Robots*, 31(4):299–316, 2011.

C.I. Connolly. Harmonic functions and collision probabilities. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3015–3019, San Diego, California, USA, 1994.

C.I. Connolly, J.B. Burns, and R. Weiss. Path planning using Laplace's equation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2102–2106, Cincinnati, Ohio, USA, 1990.

R. Cressman, V. Křivan, and J. Garay. Ideal free distributions, evolutionary games, and population dynamics in multiple-species environments. *The American Naturalist*, 164(4): 473–489, 2004.

P. Dario, M.C. Carrozza, E. Guglielmelli, C. Laschi, A. Menciassi, S. Micera, and F. Vecchi. Robotics as a future and emerging technology. *IEEE Robotics and Automation Magazine*, 2005.

R.W. Dawes. Some pursuit evasion problems on grids. *Information Processing Letters*, 43(5): 241–247, 1992.

F. de la Paz López, J.R. Álvarez-Sánchez, J.I. Rosado Sánchez, and J.M. Cuadra Troncoso. Mathematical foundations of the center of area method for robot navigation. In *Proceedings of the 3rd International Work-Conference on the Interplay Between Natural and Artificial Computation: Part II: Bioinspired Applications in Artificial and Natural Computation*, pages 419–428, Santiago de Compostela, Spain, 2009.

R. de Villiers, C.J. Wright, and Y. Yavin. A stochastic pursuit-evasion differential game in 3-D: optimal evasion strategies. *Computers & Mathematics with Applications*, 13(7):623–630, 1987.

F. Dercole, R. Ferrière, A. Gragnani, and S. Rinaldi. Coevolution of slowfast populations: evolutionary sliding, evolutionary pseudo-equilibria and complex Red Queen dynamics. *Proceedings of the Royal Society B: Biological Sciences*, 273(1589):983–990, 2006.

S.F. Desouky and H.M. Schwartz. A novel technique to design a fuzzy logic controller using Q(λ)-learning and genetic algorithms in the pursuit-evasion game. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 2609–2615, San Antonio, Texas, USA, 2009.

C. DiLeo and X. Deng. Design of and experiments on a dragonfly-inspired robot. *Advanced Robotics*, 23(7-8):1003–1021, 2009.

R. Dillmann, J. Albiez, B. Gassmann, T. Kerscher, and M. Zöllner. Biologically inspired walking machines: design, control and perception. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1850):133–151, 2007.

F. Ding, J. Peng, X. Bian, and H. Wang. AUV local path planning based on virtual potential field. In *Proceedings of the IEEE International Conference on Mechatronics and Automation*, pages 1711–1716, Niagara Falls, Canada, 2005.

B. Drossel, P.G. Higgs, and A.J. McKane. The influence of predator-prey population dynamics on the long-term evolution of food web structure. *Journal of Theoretical Biology*, 208(1):91–107, 2001.

E. Duman, M. Kaya, and E. Akin. A multi-agent fuzzy-reinforcement learning method for continuous domains. In *Proceedings of the 4th international Central and Eastern European conference on Multi-Agent Systems and Applications*, pages 306–315, Budapest, Hungary, 2005.

A. Dumitrescu, H. Kok, I. Suzuki, and P. Zylinski. Vision-based pursuit-evasion in a grid. *SIAM Journal on Discrete Mathematics*, 24(3):1177–1204, 2010.

P.A. Dunker, W.A. Lewinger, A.J. Hunt, and R.D. Quinn. A biologically inspired robot for lunar in-situ resource utilization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5039–5044, St. Louis, Missouri, USA, 2009.

H. Durrant-Whyte, D. Rye, and E. Nebot. Localisation of automatic guided vehicles. In *The 7th International Symposium of Robotics Research*, pages 613–625, Herrsching, Germany, 1996.

T. Eiter and H. Mannila. Computing discrete fréchet distance. Technical report, Technical University of Vienna, Department of Computer Science, 1994.

E. Even-Dar and Y. Mansour. Learning rates for Q-learning. *Journal of Machine Learning Research*, 5:1–25, 2003.

L. Fan, P. Dasgupta, and K. Cheng. Swarming-based mobile target following using limited-capability mobile mini-robots. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 168–175, Nashville, Tennessee, USA, 2009.

M.C.O. Ferrari, F. Messier, and D.P. Chivers. Can prey exhibit threat-sensitive generalization of predator recognition? Extending the Predator Recognition Continuum Hypothesis. *Proceedings of the Royal Society B: Biological Sciences*, 275(1644):1811–1816, 2008.

S.G. Ficici and J.B. Pollack. Statistical reasoning strategies in the pursuit and evasion domain. In *Proceedings of Advances in Artificial Life*, pages 79–88, Lausanne, Switzerland, 1999.

D. Floreano and S. Nolfi. Adaptive behavior in competing co-evolving species. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 378–387, Brighton, UK, 1997. MIT Press.

F.V. Fomin and D.M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399(3):236–245, 2008.

F.V. Fomin, P.A. Golovach, and J. Kratochvíl. On tractability of cops and robbers game. In

Proceedings of the 5th IFIP International Conference on Theoretical Computer Science, volume 273, pages 171–185, Milano, Italy, 2008.

F.V. Fomin, P.A. Golovach, J. Kratochvíl, N. Nisse, and K. Suchan. Pursuing a fast robber on a graph. *Theoretical Computer Science*, 411(7-9):1167–1181, 2010.

N. Franceschini, J.M. Pichon, and C. Blanes. From insect vision to robot vision. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 337(1281):283–294, 1992.

Z.E. Fuchs, P.P. Khargonekar, and J. Evers. Cooperative defense within a single-pursuer, two-evader pursuit evasion differential game. In *Proceedings of the 49th IEEE Conference on Deicision and Control*, pages 3091–3097, Atlanta, Georgia, USA, 2010.

T. Fujii, Y. Arai, H. Asama, and I. Endo. Multilayered reinforcement learning for complicated collision avoidance problems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2186–2191, Leuven, Belgium, 1998.

N. Furuichi. Dynamics between a predator and a prey switching two kinds of escape motions. *Journal of Theoretical Biology*, 217(2):159–166, 2002.

S. Gakkhar and R.K. Naji. Order and chaos in predator to prey ratio-dependent food chain. *Chaos, Solitons and Fractals*, 18(2):229–239, 2003.

K.S. Galloway, E.W. Justh, and P.S. Krishnaprasad. Motion camouflage in a stochastic setting. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 1652–1659, New Orleans, Louisiana, USA, 2007.

E. Garcia, M.A. Jimenez, P. Gonzalez de Santos, and M. Armada. The evolution of robotics research: from industrial robotics to field and service robotics. *IEEE Robotics and Automation Magazine*, pages 90–103, 2007.

M.R. Garvie. Finite-difference schemes for reaction-diffusion equations modeling predator-prey interactions in MATLAB. *Bulletin of Mathematical Biology*, 69(3):931–956, 2007.

S.S. Ge and Y.J. Cui. New potential functions for mobile robot path planning. *IEEE Transactions on Robotics and Automation*, 16(5):615–620, 2000.

J. Giesbrecht. Global path planning for unmanned ground vehicles. Technical report, Defence R&D Canada, Suffield, 2004.

P. Glendinning. The mathematics of motion camouflage. *Proceedings of the Royal Society B: Biological Sciences*, 271(1538):477–481, 2004.

V.Y. Glizer. Homicidal chauffeur game with target set in the shape of a circular angular sector: conditions for existence of a closed barrier. *Journal of Optimization Theory and Applications*, 101(3):581–598, 1999.

P.Y. Glorennec. Fuzzy Q-learning and dynamical fuzzy Q-learning. In *Proceedings of the 3rd IEEE Conference on Fuzzy Systems*, pages 474–479, Orlando, Florida, USA, 1994.

A. Gosavi. Reinforcement learning: a tutorial survey and recent advances. *Journal on Computing*, 21(2):178–192, 2008.

F.W. Grasso and P. Setlur. Inspiration, simulation and design for smart robot manipulators from the sucker actuation mechanism of cephalopods. *Bioinspiration and Biomimetics*, 2 (4):170–181, 2007.

X. Guan, W. Wang, and Y. Cai. Spatiotemporal dynamics of a Leslie-Gower predator-prey model incorporating a prey refuge. *Nonlinear Analysis: Real World Applications*, 12(4): 2385–2395, 2011.

J. Guldner and V.I. Utkin. Sliding mode control for gradient tracking and robot navigation using artificial potential fields. *IEEE Transactions on Robotics and Automation*, 11(2):247–254, 1995.

I. Gültekin and A. Arslan. Modular-fuzzy cooperation algorithm for multi-agent systems. In *Proceedings of the Second International Conference on Advances in Information Systems*, pages 255–263, Izmir, Turkey, 2002.

O. Gurel-Gurevich. Pursuit-evasion games with incomplete information in discrete time. *International Journal of Game Theory*, 38(3):367–376, 2009.

M.K. Habib, K. Watanabe, and K. Izumi. Biomimetics robotis from bio-inspiration to implementation. In *Proceedings of the 33rd Conference of the IEEE Industrial Electronics Society*, pages 143–148, Taipei, Taiwan, 2007.

H. Haddad, M. Khatib, S. Lacroix, and R. Chatila. Reactive navigation in outdoor environments using potential fields. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1232–1237, Leuven, Belgium, 1998.

T. Haynes and S. Sen. *Adaptation and learning in multiagent systems*, chapter Evolving Behavioral Strategies in Predators and Prey, pages 113–126. Springer Verlag, 1996.

J.P. Hespanha, M. Prandini, and S. Sastry. Probabilistic pursuit-evasion games: a one-step Nash approach. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 2432–2437, Sydney, New South Wales, Australia, 2000.

D. Hládek, J. Vaščák, and P. Sinčák. Hierarchical fuzzy inference system for robotic pursuit evasion task. In *Proceedings of the 6th International Symposium on Applied Machine Intelligence and Informatics*, pages 273–277, Herlany, Slovakia, 2008.

S. Hong-yan and S. Zhi-qiang. Study on a solution of pursuit-evasion differential game based on artificial fish school algorithm. In *Proceedings of the Chinese Control and Decision Conference*, pages 2092–2096, Xuzhou, China, 2010.

F. Huang, L. Wang, Q. Wang, M. Wu, and Y. Jia. Coordinated control of multiple mobile robots in pursuit-evasion games. In *Proceedings of the American Control Conference*, pages 2861–2866, St. Louis, Missouri, USA, 2009.

W.H. Huang, B.R. Fajen, J.R. Fink, and W.H. Warren. Visual navigation and obstacle avoidance using a steering potential function. *Robotics and Autonomous Systems*, 54(4): 288–299, 2006.

D. Huh, J. Park, U. Huh, and H. Kim. Path planning and navigation for autonomous mobile robot. In *Proceedings of the IEEE 28th Annual Conference of the Industrial Electronics Society*, pages 1538–1542, Sevilla, Spain, 2002.

N.B. Hui and D.K. Pratihar. Soft computing-based navigation schemes for a real wheeled robot moving among static obstacles. *Journal of Intelligent and Robotic Systems*, 51(3):333–368, 2008.

D.A. Humphries and P.M. Driver. Protean defence by prey animals. *Oecologia*, 5(4):285–302, 1970.

Y.K. Hwang and N. Ahuja. Gross motion planning - a survey. *ACM Computing Surveys*, 24 (3):219–291, 1992a.

Y.K. Hwang and N. Ahuja. A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*, 8(1):23–32, 1992b.

R. Isaacs. *Differential games*. Dover, 1965.

M. Ishikawa, T. Hagiwara, N. Yamamoto, and F. Kiriake. Brain-inspired emergence of behaviors in mobile robots by reinforcement learning with internal rewards. In *Proceedings of the 8th International Conference on Hybrid Intelligent Systems*, pages 138–143, Barcelona, Spain, 2008.

Y. Ishiwaka, T. Sato, and Y. Kakazu. An approach to the pursuit problem on a heterogeneous multiagent system using reinforcement learning. *Robotics and Autonomous Systems*, 43(4):245–256, 2003.

V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *IEEE Transactions on Robotics*, 21(5):875–884, 2005.

V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion with local visibility. *SIAM Journal on Discrete Mathematics*, 20(1):26–41, 2006.

A.R. Ives, B.J. Cardinale, and W.E. Snyder. A synthesis of subdisciplines: predator-prey interactions, and biodiversity and ecosystem functioning. *Ecology Letters*, 8(1):102–116, 2005.

F. Janabi-Sharifi and D. Vinke. Robot path planning by integrating the artificial potential field approach with simulated annealing. In *Proceedings of the International Conference on Systems, Man and Cybernetics*, pages 282–287, Le Touquet, France, 1993.

M.A.K. Jaradat, M. Al-Rousan, and L. Quadan. Reinforcement based mobile robot navigation in dynamic environment. *Robotics and Computer-Integrated Manufacturing*, 27 (1):135–149, 2011.

R.A. Jarvis and M.S. Marzouqi. Robot path planning in high risk fire front environments. In *Proceedings of the IEEE Region 10 Conference*, pages 665–670, Melbourne, Australia, 2005.

I. Jeong and J. Lee. Evolving fuzzy logic controllers for multiple mobile robots solving a continuous pursuit problem. In *Proceedings of the IEEE International Fuzzy Systems Conference*, pages 685–690, Seoul, South Korea, 1999.

J.M. Jeschke and R. Tollrian. Prey swarming: which predators become confused and why? *Animal Behaviour*, 74(3):387–393, 2007.

S. Jin and Z. Qu. Pursuit-evasion games with multi-pursuer vs. one fast evader. In *Proceedings of the 8th World Congress on Intelligent Control and Automation*, pages 3184–3189, Jinan, China, 2010.

X. Jing and Y. Wang. Local minima-free design of artificial coordinating fields. *Journal of Control Theory and Applications*, 2(4):371–380, 2004.

J. Johnson and P. Picton. *Concepts in Artificial Intelligence*, volume 2. Butterworth-Heineman LTD, 1994.

E.W. Justh and P.S. Krishnaprasad. Steering laws for motion camouflage. *Proceedings of the Royal Society A: Mathematical Physical and Engineering Sciences*, 462(2076):3629–3643, 2006.

K. Kamei and M. Ishikawa. A genetic approach to optimizing the values of parameters in reinforcement learning for navigation of a mobile robot. In *Proceedings of the International Conference on Neural Information Processing*, pages 1148–1153, Calcutta, India, 2004.

K. Kamei and M. Ishikawa. Dependency of parameter values in reinforcement learning for navigation of a mobile robot on the environment. *Neural Information Processing-Letters and Reviews*, 10(8-9):219–226, 2006.

S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for a class of pursuit-evasion games. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, Singapore, Singapore, 2010.

N. Karnad and V. Isler. Bearing-only pursuit. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2665–2670, Pasadena, California, USA, 2008.

M. Kaya and R. Alhajj. Reinforcement learning in multiagent systems: a modular fuzzy approach with internal model capabilities. In *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence*, pages 469–474, Washington, D.C., USA, 2002.

M. Kaya and R. Alhajj. Multiagent reinforcement learning using OLAP-based association rules mining. In *Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology*, pages 584–587, Halifax, Nova Scotia, Canada, 2003.

M. Kaya and R. Alhajj. Modular fuzzy-reinforcement learning approach with internal model capabilities for multiagent systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 34(2):1210–1223, 2004.

S. Kernbach, R. Thenius, O. Kernbach, and T. Schmickl. Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system. *Adaptive Behavior*, 17(3):237–259, 2009.

A.D. Khalafi and M.R. Toroghi. Capture zone in the herding pursuit evasion games. *Applied Mathematical Sciences*, 5(39):1935–1945, 2011.

O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

P. Khosla and R. Volpe. Superquadric artificial potentials for obstacle avoidance and approach. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 1778–1784, Philadelphia, Philadelphia, USA, 1988.

A. Kilic and A. Arslan. Min-max fuzzy Q-learning in cooperative multi-agent systems. In *Proceedings of the Second International Conference on Advances in Information Systems*, pages 264–272, Izmir, Turkey, 2002.

H.J. Kim and D.H. Shim. A flight control system for aerial robots: algorithms and experiments. *Control Engineering Practice*, 11(12):1389–1400, 2003.

H.J. Kim, R. Vidal, D.H. Shim, O. Shakernia, and S. Sastry. A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles. In *Proceedings of the 40th Conference on Decision and Control*, pages 634–639, Orlando, Florida, USA, 2001.

J. Kim and P.K. Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, 8(3):338–349, 1992.

J. Kim and M. Tahk. Co-evolutionary computation for constrained min-max problems and its applications for pursuit-evasion games. In *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1205–1212, Seoul, South Korea, 2001.

H. Kimura, Y. Fukouka, and A.H. Cohen. Biologically inspired adaptive walking of a quadruped robot. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1850):153–170, 2007.

K. Klein and S. Suni. Multiagent pursuit evasion, or playing kabaddi. *Algorithmic Foundations of Robotics IX*, 68:89–104, 2010.

D.E. Koditschek. Exact robot navigation by means of potential functions: Exact robot navigation by means of potential functions: Some topological considerations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 1–6, Raleigh, North Carolina, USA, 1987.

S. Kopparty and C.V. Ravishankar. A framework for pursuit evasion games in r^n. *Information Processing Letters*, 96(3):114–122, 2005.

Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1398–1404, Sacramento, California, USA, 1991.

D.J. Kwak and H.J. Kim. Probability map partitioning for multi-player pursuit-evasion game. In *Proceedings of the International Conference on Control, Automation and Systems*, pages 294–298, Gyeonggi-do, South Korea, 2010.

R. Lachner, M.H. Breitner, and H.J. Pesch. *Variational Calculus, Optimal Control and Applications*, volume 124, chapter Real-time computation of strategies of differential games with applications to collision avoidance, pages 281–290. Birkhäuser, 1998.

A. Laud and G. DeJong. The influence of reward on the speed of reinforcement learning: An analysis of shaping. In *Proceedings of the 20th International Conference on Machine Learning*, pages 440–447, Washington, D.C., USA, 2003.

S.M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

J. Lee, Y. Nam, and S. Hong. Random force based algorithm for local minima escape of potential field method. In *Proceedings of the International Conference on Control, Automation, Robotics and Vision*, pages 827–832, Singapore, Singapore, 2010.

S.W. Lee. A bio-inspired group evasion behaviour. Technical report, Department of Computer Science, The University of North Carolina at Chapel Hill, 2008.

C.T. Leondes and B. Mons. On-line solution of a stochastic pursuit-evasion game. *Journal of Optimization Theory and Applications*, 28(3), 1979.

D. Li and J.B. Cruz. Improvement with look-ahead on cooperative pursuit games. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 4313–4318, San Diego, California, USA, 2006.

D. Li and J.B. Cruz. A two-player stochastic pursuit-evasion differential game. In *Proceedings of the 46th Conference on Decision and Control*, pages 4623–4628, New Orleans, Louisiana, USA, 2007.

D. Li, J.B. Cruz, and C.J. Schumacher. Stochastic multi-player pursuit-evasion differential games. *International Journal of Robust and Nonlinear Control*, 18(2):218–247, 2007.

C. Liang, M. Ceccarelli, and G. Carbone. A novel biologically inspired tripod walking robot. In *Proceedings of the 13th WSEAS International Conference on Systems*, pages 83–91, Rodos Island, Greece, 2009.

S.H. Lim, T. Furukawa, G. Dissanayake, and H.F. Durrant-Whyte. A time-optimal control strategy for pursuit-evasion games problems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3962–3967, New Orleans, Louisiana, USA, 2004.

Y. Lipman, J. Shinar, and Y. Oshman. Stochastic analysis of the interception of maneuvering antisurface missiles. *Journal of Guidance, Control, and Dynamics*, 20(4):707–714, 1997.

P. Liu and Z. Jin. Pattern formation of a predator-prey model. *Nonlinear Analysis: Hybrid Systems*, 3(3):177–183, 2009.

R.J. Lock, R. Vaidyanathan, S.C. Burgess, and J. Loveless. Development of a biologically inspired multi-modal wing model for aerial-aquatic robotic vehicles through empirical and numerical modelling of the common guillemot, *Uria aalge*. *Bioinspiration and Biomimetics*, 5(4):1–15, 2010.

C. Louste and A. Liegeois. Near optimal robust path planning for mobile robots: the viscous fluid method with friction. *Journal of Intelligent and Robotic Systems*, 27(1-2):99–112, 2000.

K.H. Low, C. Zhou, T.W. Ong, and J. Yu. Modular design and initial gait study of an amphibian robotic turtle. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, pages 535–540, Bangkok, Thailand, 2008.

S. Luke. *Essentials of Metaheuristics*. Lulu, 2009.

C. Luo and S.X. Yang. A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments. *IEEE Transactions on Neural Networks*, 19(7):1279–1298, 2008.

Z. Luo, Q. Cao, and Y. Zhao. A self-adaptive predictive policy for pursuit-evasion game. *Journal of Information Science and Engineering*, 24(5):1397–1407, 2008.

B. Marthi. Automatic shaping and decomposition of reward functions. In *Proceedings of the 24th International Conference on Machine Learning*, pages 607–608, Corvallis, Oregon, USA, 2007.

A. Martinoli, A.J. Ijspeert, and F. Mondada. Understanding collective aggregation mechanisms: from probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems*, 29(1):51–63, 1999.

M. Marzouqi and R.A. Jarvis. Covert path planning in unknown environments with known or suspected sentry locations. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1-4, pages 3320–3326, Edmonton, Alberta, Canada, 2005.

M.S. Marzouqi and R.A. Jarvis. New visibility-based path-planning approach for covert robotic navigation. *Robotica*, 24(6):759–773, 2006.

E. Masehian and M.R. Amin-Naseri. A Voronoi diagram-visibility graph-potential field compound algorithm for robot path planning. *Journal of Robotic Systems*, 21(6):275–300, 2004.

A. Masoud. Evasion of multiple pursuers in a stationary, cluttered, environment: a harmonic potential field approach. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 654 – 659, Vancouver, British Columbia, Canada, 2002.

A.A. Masoud. Using hybrid vector-harmonic potential fields for multi-robot, multi-target navigation in a stationary environment. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3564–3571, Minneapolis, Minnesota, USA, 1996.

A.A. Masoud. Evasion of multiple, intelligent pusuers in a stationary, cluttered environment using a Poisson potential field. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4234–4239, Taipei, Taiwan, 2003a.

A.A. Masoud. Escaping capture by multiple, intelligent, well-informed, cooperative pursuers amidst stationary clutter. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, Maui, Hawaii, USA, 2003b.

A.A. Masoud. Decentralized, self-organizing, potential field-based control for individually-motivated, mobile agents in a cluttered environment: a vector-harmonic potential field approach. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(3):372 – 390, 2007a.

A.A. Masoud. Dynamic trajectory generation for spatially constrained mechanical systems using harmonic potential fields. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1980–1985, Roma, Italy, 2007b.

A.A. Masoud. Managing the dynamics of a harmonic potential field-guided robot in a cluttered environment. *Transactions on Industrial Electronics*, 56(2):488–496, 2009.

A.A. Masoud. Motion planning with gamma-harmonic potential fields. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 297–302, Montreal, Ontario, Canada, 2010.

A.A. Masoud, S.A. Masoud, and M.M. Bayoumi. Robot navigation using a pressure generated mechanical stress field "the biharmonic potential approach". In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 124–129, San Diego, California, USA, 1994.

S.A. Masoud and A.A. Masoud. Constrained motion control using vector potential fields. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 30(3):251–272, 2000.

S.A. Masoud and A.A. Masoud. Motion planning in the presence of directional and regional avoidance constraints using nonlinear, anisotropic, harmonic potential fields: Motion planning in the presence of directional and regional avoidance constraints using nonlinear, anisotropic, harmonic potential fields: a physical metaphor. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 32(6):705–723, 2002.

M.J. Matarić. Reward functions for accelerated learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 181–189, New Brunswick, New Jersey, USA, 1994.

M.J. Matarić. Learning in behaviour-based multi-robot systems: policies, models, and other agents. *Cognitive Systems Research*, 2(1):81–93, 2001.

J.B. Mbede, P.Ele, and H. Xinhan. Neuro-fuzzy dynamic obstacle avoidance for autonomous robot manipulators. *JSME International Journal Series C*, 45(1):286–297, 2002.

R. Mchich, P. Auger, and C. Lett. Effects of aggregative and solitary individual behaviors on the dynamics of predator-prey game models. *Ecological Modelling*, 197(3-4):281–289, 2006.

C.R. McInnes. Velocity field path-planning for single and multiple unmanned aerial vehicles. *Aeronautical Journal*, 107(1073):419–426, 2003.

K.A. McIsaac, J. Ren, and X. Huang. Modified Newton's method applied to potential field navigation. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 5873–5878, Maui, Hawaii, USA, 2003.

D.B. Megherbi and V. Malayia. A hybrid cognitive/reactive intelligent agent autonomous path planning technique in a networked-distributed unstructured environment for reinforcement learning. *The Journal of Supercomputing*, 59(3):1188–1217, 2012.

L. Meier. A new technique for solving pursuit-evasion differential games. *IEEE Transactions on Automatic Control*, 14(4):352–359, 1969.

F.S. Melo and M.I. Ribeiro. Reinforcement learning with function approximation for cooperative navigation tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3321–3327, Pasadena, California, USA, 2008.

X. Meng, R. Babuška, L. Buşoniu, Y. Chen, and W. Tan. An improved multiagent reinforcement learning algorithm. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 337–343, Compiegne, France, 2005.

M.E. Mendoza Meza, A. Bhayaa, E. Kaszkurewicza, and M. Iskin da Silveira Costa. Threshold policies control for predator-prey systems using a control Lyapunov function approach. *Theoretical Population Biology*, 67(4):273–284, 2005.

G.F. Miller. *Machiavellian intelligence II: extensions and evaluations*, chapter Protean Primates: The Evolution of Adaptive Unpredictability in Competition and Courtship, pages 312–340. Cambridge U. Press, 1997.

G.F. Miller and D. Cliff. Co-evolution of pursuit and evasion I: biological and game-theoretic foundations. Technical report, School of Cognitive and Computing Sciences, University of Sussex, 1994.

T. Miloh and M. Pachter. Ship collision-avoidance and pursuit-evasion differential games with speed-loss in a turn. *Computers and Mathematics with Applications*, 18(1-3):77–100, 1989.

R.S. Mirza, M.C.O. Ferrari, J.M. Kiesecker, and D.P. Chivers. Responses of American toad tadpoles to predation cues: behavioural response thresholds, threat-sensitivity and acquired predation recognition. *Behaviour*, 143(7):877–889, 2006.

W.A. Mitchell. Multi-behavioral strategies in a predator-prey game: an evolutionary algorithm analysis. *Oikos*, 118(7):1073–1083, 2009.

N. Mitsumoto, T. Fukuda, K. Shimojima, and A. Ogawa. Micro autonomous robotic system and biologically inspired immune swarm strategy as multiagent robotic system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2187–2192, Nagoya, Japan, 1995.

K. Mizukami and K. Eguchi. A geometrical approach to problems of pursuit-evasion games. *Journal of The Franklin Institute-Engineering and Applied Mathematics*, 303(4):371–384, 1977.

F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.C. Zufferey, D. Floreano, and A. Martinoli. The e-Puck, a robot designed for education in engineering. Technical report, École Polytechnique Fédérale de Lausanne, 2009.

M.C. Mora and J. Tornero. Path planning and trajectory generation using multi-rate predictive artificial potential fields. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2990–2995, Nice, France, 2008.

K. Morihiro, H. Nisimura, T. Isokawa, and N. Matsui. Reinforcement learning scheme for grouping and anti-predator behavior. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 115–122. Springer Berlin / Heidelberg, 2007.

P.J. Morin. *Community Ecology*. Wiley-Blackwell, 1999.

R. Murrieta-Cid, T. Muppirala, A. Sarmiento, S. Bhattacharya, and S. Hutchinson. Surveillance strategies for a pursuer with finite sensor range. *International Journal of Robotics Research*, 26(3):233–253, 2007.

M.G. Neubert, M. Kot, and M.A. Lewis. Dispersal and pattern-formation in a discrete-time predator-prey model. *Theoretical Population Biology*, 48(1):7–43, 1995.

S.W. Neufeld. A pursuit-evasion problem on a grid. *Information Processing Letters*, 58(1): 5–9, 1996.

J. Ni and S.X. Yang. Bioinspired neural network for real-time cooperative hunting by multirobots in unknown environments. *IEEE Transactions on Neural Networks*, 22(12): 2062–2077, 2011.

N. Nisse and K. Suchan. *Graph-Theoretic Concepts in Computer Science*, chapter Fast robber in planer graphs, pages 312–323. Springer Verlag Berlin / Heidelberg, 2008.

G. Nitschke. Emergence of specialized behavior in a pursuit-evasion game. In *Proceedings of the International Central and Eastern European Conference on Multi-Agent Systems*, pages 324–334, Prague, Czech Republic, 2003.

S. Nolfi and D. Floreano. Coevolving predator and prey robots: Do "arms races" arise in artificial evolution? *Artificial Life*, 4(4):311–335, 1999.

J.M. Norman. From cave paintings to the internet, chronological and thematic studies on the history of information and media, 2004-2011. URL `http://www.historyofinformation.com/index.php`.

T. Oboshi, S. Kato, A. Mutoh, and H. Itoh. *Artificial Life VIII*, chapter Collective or scattering: evolving schooling behaviors to escape from predator, pages 386–389. MIT Press, 2002.

N. Ohnishi and A. Imiya. *Robot Vision*, chapter Visual Navigation of Mobile Robot Using Optical Flow and Visual Potential Field, pages 412–426. Springer-Verlag Berlin Heidelberg, 2008.

J. Or. A hybrid CPG-ZMP control system for stable walking of a simulated flexible spine humanoid robot. *Neural Networks*, 23(3):452–460, 2010.

M. Pachter and Y. Yavin. A stochastic homicidal chauffeur pursuit-evasion differential game. *Journal of Optimization Theory and Applications*, 34(3):405–424, 1981.

M. Pachter and Y. Yavin. Simple-motion pursuit-evasion differential games, part 1:

sroboscopic strategies in collision-course guidance and proportional navigation. *Journal of Optimization Theory and Applications*, 51(1):95–127, 1986.

L. Panait and S. Luke. Cooperative multi-agent learning: the state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

J. Park, J. Choi, J. Kim, and B. Lee. Roadmap-based stealth navigation for intercepting an invader. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 442–447, Kobe, Japan, 2009.

J. Park, J.S. Choi, J. Kim, S. Ji, and B.H. Lee. Long-term stealth navigation in a security zone where the movement of the invader is monitored. *International Journal of Control Automation and Systems*, 8(3):604–614, 2010.

M.G. Park and M.C. Lee. A new technique to escape local minimum in artificial potential field based path planning. *Journal of Mechanical Science and Technology*, 17(12):1876–1885, 2003.

I. Partalas, I. Feneris, and I. Vlahavas. Multi-agent reinforcement learning using strategies and voting. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, pages 318–324, Patras, Greece, 2007.

K. Pathak and S.K. Agrawal. Planning and control of a nonholonomic unicycle using ring shaped local potential fields. In *Proceedings of the American Control Conference*, pages 2368–2373, Boston, Massachusetts, USA, 2004.

M.J. Pearson, A.G. Pipe, C. Melhuish, B. Mitchinson, and T.J. Prescott. Whiskerbot: a robotic active touch system modeled on the rat whisker sensory system. *Adaptive Behavior*, 15(3): 223–240, 2007.

M. Peeters, K. Verbeeck, and A. Nowé. The effect of bootstrapping in multi-automata reinforcement learning. In *Proceedings of the IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 76–83, Honolulu, Hawaii, USA, 2007.

H.J. Pesch, I. Gabler, S. Miesbach, and M.H. Breitner. *New Trends in Dynamic Games and Applications*, chapter Synthesis of optimal strategies for differential games by neural networks, pages 111–141. Birkhäuser, 1995.

S.V. Petrovskii and H. Malchow. A minimal model of pattern formation in a prey-predator system. *Mathematical and Computer Modelling*, 29(8):49–63, 1999.

R. Pfeifer, M. Lungarella, and F. Iida. Self-organization, embodiment, and biologically inspired robotics. *Science*, 318(5853):1088–1093, 2007.

L.C.A. Pimienta, G.A.S. Pereira, R.C. Mesquita, A.R. Fonseca, E.J. Silva, W.M. Caminhas, and M.F.M. Campos. Robot navigation based on electrostatic field computation. *IEEE Transactions on Magnetics*, 42(4):1459–1462, 2006.

P. Ponce-Cruz and F.D. Ramírez-Figueroa. *Intelligent control systems with LabVIEW*. Springer, 2009.

A. Poty, P. Melchior, and A. Oustaloup. Dynamic path planning for mobile robots using fractional potential field. In *Proceedings of the First International Symposium on Control, Communications and Signal Processing*, pages 557–561, Limassol, Cyprus, 2004.

N. Pradhan, T. Burg, and S. Birchfield. Robot crowd navigation using predictive position fields in the potential function framework. In *Proceedings of the American Control Conference*, pages 4628–4633, San Francisco, California, USA, 2011.

E. Prestes, M.A.P. Idiart, P.M. Engel, and M. Trevisan. Exploration technique using potential fields calculated from relaxation methods. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, Hawaii, USA, 2001.

F. Quian and H. Hirata. Q-learning automaton. In *Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology*, pages 432–237, Halifax, Nova Scotia, Canada, 2003.

D. Raabe, K. Alemzadeh, A.L. Harrison, and A.J. Ireland. The chewing robot: a new biologically-inspired way to evaluate dental restorative materials. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6050–6053, Minneapolis, Minnesota, USA, 2009.

T. Raivio and H. Ehtamo. Applying nonlinear programming to a complex pursuit-evasion problem. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 1548–1551, Orlando, Florida, USA, 1997.

S. Ravela, R. Weiss, B. Draper, B. Pinette, A. Hanson, and E. Riseman. Stealth navigation: planning and behaviors. In *Proceedings of the ARPA Image Understanding Workshop*, pages 1093–1100, Monterey, California, USA, 1994.

J. Ren, K.A. McIsaac, R.V. Patel, and T.M. Peters. A potential model using generalized sigmoid functions. *IEEE Transactions on Systems, Man, and Cybernetics Part B-Cybernetics*, 37(2):477–484, 2007.

D. Renjewski, A. Seyfarth, P. Manoonpong, and F. Woergoetter. The development of a biomechanical leg system and its neural control. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, pages 1894–1899, Guilin, China, 2009.

E. Rimon and D.E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992.

M.B. Ring. *Continual learning in reinforcement environments*. PhD thesis, The University of Texas at Austin, 1994.

R. Robotin, G. Lazea, and C. Marcu. *Engineering the Future*, chapter Graph search techniques for mobile robot path planning, pages 159–178. InTechOpen, 2010.

A. Roennau, T. Kerscher, and R. Dillmann. Design and kinematics of a biologically-inspired leg for a six-legged walking machine. In *Proceedings of the 3rd IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 626–631, Tokyo, Japan, 2010.

G. Roussos, D.V. Dimarogonas, and K.J. Kyriakopoulos. 3d navigation and collision avoidance for nonholonomic aircraft-like vehicles. *International Journal of Adaptive Control and Signal Processing*, 24(10):900–920, 2010.

S.J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prenticel Hall, 2003.

J.C. Santamaria, R.S., and A. Ram. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, 6(2):163–217, 1997.

J.E. Satulovsky and T. Tomé. Stochastic lattice gas model for a predator-prey system. *Physical Review E*, 49(6):5073–5079, 1994.

L. Schenato, S. Oh, S. Sastry, and P. Bose. Swarm coordination for pursuit evasion games using sensor networks. In *Proceedings of the International Conference on Robotics and Automation*, pages 2493–2498, Barcelona, Spain, 2005.

A. Scott and U. Stege. Parametrized pursuit-evasion games. *Theoretical Computer Science*, 411(43):3845–3858, 2010.

N.E. Sharkey. *Handbook of Brain Theory and Neural Networks*, chapter Biologically inspired robotics. MIT Press, 2002.

A.G. Shem, T.A. Mazzuchi, and S. Sarkani. Addressing uncertainty in UAV navigation decision-making. *IEEE Transactions on Aerospace and Electronic Systems*, 44(1):295–313, 2008.

C. Shi, M. Zhang, and J. Peng. Harmonic potential field method for autonomous ship navigation. In *Proceedings of the 7th International Conference on ITS Telecommunications*, pages 471–476, Sophia Antipolis, France, 2007.

T. Shima and J. Shinar. Time-varying linear pursuit-evasion game models with bounded controls. *Journal of Guidance, Control, and Dynamics*, 25(3):425–432, 2002.

J. Shinar and T. Shima. A game theoretical interceptor guidance law for ballistic missile defense. In *Proceedings of the 35th Conference on Decision and Control*, pages 2780–2785, Kobe, Japan, 1996.

J. Shinar and T. Shima. Nonorthodox guidance law development approach for intercepting maneuvering targets. *Journal of Guidance, Control, and Dynamics*, 25(4):658–666, 2002.

J. Shinar and G. Silberman. A discrete dynamic game modelling anti-missile defense scenarios. *Dynamics and Control*, 5(1):55–67, 1995.

J. Shinar, V.Y. Glizer, and V. Turetsky. Robust pursuit of a hybrid evader. *Applied Mathematics and Computation*, 217(3):1231–1245, 2010.

B. Siciliano and O. Khatib, editors. *Springer Handbook of Robotics*. Springer-Verlag Berlin Heidelberg, 2008.

K. Sigmund. William D. Hamilton's work in evolutionary game theory. *Theoretical Population Biology*, 59(1):3–6, 2001.

R. Silveira, E. Prestes, and L. Nedel. Fast path planning using multi-resolution boundary value problems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4710–4715, Taipei, Taiwan, 2010.

M.G. Slack. Navigation templates: mediating qualitative guidance and quantitative control in mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(2):452–466, 1993.

W.D. Smart and L. Pack Kaelbling. Effective reinforcement learning for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3404–3410, Washington, D.C., USA, 2002.

Y.S. Song and M. Sitti. Surface-tension-driven biologically inspired water strider robots: theory and experiments. *IEEE Transactions on Robotics*, 23(3):578–589, 2007.

L. Southard, T.M. Hoeg, D.W. Palmer, J. Antol, R.M. Kolacinski, and R.D. Quinn. Exploring Mars using a group of tumbleweed rovers. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 775–780, Roma, Italy, 2007.

L. Sowerby. *Vector field theory with applications*. Longman Group Limited, 1974.

M.V. Srinivasan, S. Thurrowgood, and D. Soccol. Competent vision and navigation systems from flying insects to autonomously navigating robots. *IEEE Robotics and Automation Magazine*, 16(3):59–71, 2009.

T. Stankowich and D.T. Blumstein. Fear in animals: a meta-analysis and review of risk assessment. *Proceedings of the Royal Society B: Biological Sciences*, 272(1581):2627–2634, 2005.

F.B. Sumner. Studies of protective color change III. Experiments with fish both as predators and prey. *Proceedings of the National Academy of Scienes of the United States of America*, 21 (6):345–353, 1935.

R. Sun, S. Tatsumi, and G. Zhao. Q-ac: multiagent reinforcement learning with perception-conversion action. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 2950–2955, Washington, D.C., USA, 2003.

J. Sungho. Adaptive biomimetic control of robot arm motions. *Neurocomputing*, 71(16-18): 3625–3630, 2008.

R.S. Sutton and A.G. Barto. *Reinforcement learning: an introduction*. The MIT Press, 1998.

I. Suzuki and P. Zyliński. Capturing an evader in a building-randomized and deterministic algorithms for mobile robots. *IEEE Robotics & Automation Magazine*, 15(2):16–26, 2008.

C. Szepesvári. *Algorithms for reinforcement learning*. Morgan & Claypool, 2010.

M. Takahashi, T. Suzuki, H. Shitamoto, T. Moriguchi, and K. Yoshida. Developing a mobile robot for transport applications in the hospital domain. *Robotics and Autonomous Systems*, 58(7):889–899, 2010.

H. Tamakoshi and S. Ishii. Multiagent reinforcement learning applied to a chase problem in a continuous world. *Artificial Life and Robotics*, 5(4):202–206, 2001.

F. Tan, J. Yang, J. Huang, T. Jia, W. Chen, and J. Wang. A navigation system for family indoor monitor mobile robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5978–5983, Taipei, Taiwan, 2010.

J.L. Tangorra, S.N. Davidson, I.W. Hunter, P.G.A. Madden, G.V. Lauder, H. Dong, M. Bozkurttas, and R. Mittal. The development of a biologically inspired propulsor for unmmaned underwater vehicles. *IEEE Journal of Oceanic Engineering*, 32(3):533–550, 2007.

J.C. Tay, C.H. Tng, and C.S. Chan. Environmental effects on the coevolution of pursuit and evasion strategies. *Genetic Programming and Evolvable Machines*, 9(1):5–37, 2008.

S. ten Hagen and B. Kröse. Q-learning for systems with continuous state and action spaces. In *Proceedings of the 10th Belgian-Dutch Conference on Machine Learning*, Tilburg, Netherlands, 2000.

Y.A. Teng, D. DeMenthon, and L.S. Davis. Stealth terrain navigation. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1):96–110, 1993.

A. Tews, M.J. Matarić, and G.S. Sukhatme. Avoiding detection in a dynamic environment. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3773–3778, San Jose, California, USA, 2004a.

A.D. Tews, G.S. Sukhatme, and M.J. Matarić. A multi-robot approach to stealthy navigation in the presence of an observer. In *Proceedings of the International Conference on Robotics and Automation*, pages 2379–2385, New Orleans, Louisiana, USA, 2004b.

M. Trevisan, M.A.P. Idiart, E. Prestes, and P.M. Engel. Exploratory navigation based on dynamical boundary value problems. *Journal of Intelligent and Robotic Systems*, 45(2):101–114, 2006.

J. Tschirhart. A new adaptive system approach to predator-prey modeling. *Ecological Modelling*, 176(3-4):255–276, 2004.

N.C. Tsourveloudis, K.P. Valavanis, and T. Hebert. Autonomous vehicle navigation utilizing electrostatic potential fields and fuzzy logic. *IEEE Transactions on Robotics and Automation*, 17(4):490–497, 2001.

E. Uchibe and M. Asada. Incremental coevolution with competitive and cooperative tasks in a multirobot environment. *Proceedings of the IEEE*, 94(7):1412–1424, 2006.

P. Vadakkepat, T.H. Lee, and L. Xin. Application of evolutionary artificial potential field in robot soccer system. In *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, pages 2781–2785, Vancouver, British Columbia, Canada, 2001.

J. Vaščák. Navigation of mobile robots using potential fields and computational intelligence means. *Acta Polytechnica Hungarica*, 4(1):63–74, 2007.

T. Veen, D.S. Richardson, K. Blaakmeer, and J. Komdeur. Experimental evidence for innate predator recognition in the Seychelles warbler. *Proceedings of the Royal Society B: Biological Sciences*, 267(1459):2253–2258, 2000.

J. Velagic, B. Lacevic, and N. Osmic. Efficient path planning algorithm for mobile robot navigation with a local minima problem solving. In *Proceedings of the IEEE International Conference on Industrial Technology*, pages 2695–2700, Mumbai, India, 2006.

R. Vidal, S. Rashid, C. Sharp, O. Shakernia, J. Kim, and S. Sastry. Pursuit-evasion games with unmanned ground and aerial vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2948–2955, Seoul, South Korea, 2001.

R. Vidal, O. Shakernia, H.J. Kim, D.H. Shim, and S. Sastry. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 18(5):662–669, 2002.

M.A.M. Vieira, R. Govindan, and G.S. Sukhatme. Scalable and practical pursuit-evasion with networked robots. *Intelligent Service Robotics*, 2(4):247–263, 2009.

N. Vlassis. *A concise introduction to multi-agent systems and distributed artificial intelligence*. Morgan & Claypool, 2007.

M. Wahde and M.G. Nordahl. Evolution of protean behavior in pursuit-evasion contests. In *From animals to animats 5*, pages 557–561. MIT Press, 1998a.

M. Wahde and M.G. Nordahl. Coevolving pursuit-evasion strategies in open and confined regions. In *Artificial Life VI*, pages 472–476, 1998b.

L.C. Wang, L.S.Y., and M.H. Ang. Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 821–826, Vancouver, British Columbia, Canada, 2002.

C.W. Warren. Global path planning using artificial potential fields. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 316–321, Scottsdale, Arizona, USA, 1989.

T. Watanabe and Y. Takahashi. Hierarchical reinforcement learning using a modular fuzzy model for multi-agent problem. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 1681–1686, Montreal, Quebec, Canada, 2007.

M. Weir, A. Buck, and J. Lewis. POTBUG: a mind's eye approach to providing BUG-like guarantees for adaptive obstacle navigation using dynamic potential fields. In S. Nolfi, G. Baldassarre, R. Calabretta, J.C.T. Hallam, D. Marocco, J.A. Meyer, O. Miglino, and D. Parisi, editors, *From animals to animats 9*, pages 239–250, 2006.

M.K. Weir and M.P. Bott. High quality goal connection for nonholonomic obstacle navigation allowing for drift using dynamic potential fields. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3221–3226, Anchorage, Alaska, USA, 2010.

R. Weiss. *Multiagent systems: a modern approach to distributed artificial intelligence*. The MIT Press, 1999.

R.J. Williams and N.D. Martinez. Stabilization of chaotic and non-permanent food web dynamics. *The European Physical Journal B*, 38(2):297–303, 2004.

B. Wong and M. Spetsakis. Scene reconstruction and robot navigation using dynamic fields. *Autonomous Robots*, 8(1):71–86, 2000.

A. Woo, P. Poulin, and A. Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, 1990.

R.J. Wood. The first takeoff of a biologically inspired at-scale robotic insect. *IEEE Transactions on Robotics*, 24(2):341–347, 2008.

M.J. Wooldridge. *Multi-agent systems: an introduction*. Wiley, 2002.

W.L. Xu, J.S. Pap, and J. Bronlund. Design of a biologically inspired parallel robot for foods chewing. *IEEE Transactions on Industrial Electronics*, 55(2):832–841, 2008.

C.D. Yang and C.C. Yang. Analytical solution of 3D True Proportional Navigation. *IEEE Transactions on Aerospace and Electronic Systems*, 19(3):569–577, 1996.

E. Yang and D. Gu. Multiagent reinforcement learning for multi-robot systems: a survey. Technical report, Department of Computer Science, University of Essex, 2004.

G. Yang, E. Chen, and C. An. Mobile robot navigation using neural Q-learning. In *Proceedings of the 3rd International Conference on Machine Learning and Cybernetics*, pages 48–52, Shanghai, China, 2004.

Y. Yavin. A pursuit-evasion differential game with noisy measurements of the evader's bearing frm the pursuer. *Journal of Optimization Theory and Applications*, 51(1):161–177, 1986.

Y. Yavin. Pursuit-evasion differential games with deception or interrupted observation. *Computers and Mathematics with Applications*, 13(1-3):191–203, 1987.

Y. Yavin. Stochastic two-target pursuit-evasion differential games in the plane. *Journal of Optimization Theory and Applications*, 56(3):325–, 1988.

X. Yuan and S.X. Yang. Guest editorial: robotics and automation with biologically inspired intelligence. *Intelligent Automation and Soft Computing*, 15(2):127–130, 2009.

X. Yun and K. Tan. A wall-following method for escaping local minima in potential field based motion planning. In *Proceedings of the International Conference on Advanced Robotics*, pages 421–426, Monterey, California, USA, 1997.

J.S. Zelek. Dynamic path planning. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, pages 1285–1290, Vancouver, British Columbia, Canada, 1995.

D. Zhang and Z. Gao. Hybrid head mechanism of the groundhog-like mine rescue robot. *Robotics and Computer-Integrated Manufacturing*, 27(2):460–470, 2011.

D. Zhao and W. Jin. The study of cooperative behavior in predator-prey problem of multi-agent systems. In *Proceedings of Autonomous Decentralized Systems*, pages 90–96, Chengdu, China, 2005.

W. Zhao, Y. Hu, L. Zhang, and L. Wang. Design and CPG-based control of biomimetic robotic fish. *IET Control Theory and Applications*, 3(3):281–293, 2009.

J. Zheng, H. Yu, W. Liang, and P. Zeng. Probabilistic strategies to coordinate multiple robotic pursuers in pursuit-evasion games. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, pages 559–564, Sanya, China, 2007.

M. Zheng, Y. Kashimori, O. Hoshino, K. Fujita, and T. Kambara. Behavior pattern (innate action) of individuals in fish schools generating efficient collective evasion from predation. *Journal of Theoretical Biology*, 235(2):153–167, 2005.

J. Zhu, T. Zhang, and J. Song. An improved wall following method for escaping from local minimum in artificial potential field based path planning. In *Proceedings of the Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, pages 6017–6022, Shanghai, China, 2009.

Q. Zhu, Y. Yan, and Z. Xing. Robot path planning based on artificial potential field approach with simulated annealing. In *Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications*, pages 622 – 627, Jinan, China, 2006.

Q.M. Zhu. Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation. *IEEE Transactions on Robotics and Automation*, 7(3):390–397, 1991.

X. Zou and J. Zhu. Virtual local target method for avoiding local minimum in potential field based robot navigation. *Journal of Zhejiang University SCIENCE*, 4(3):264–269, 2003.