Constructing informative Bayesian priors to improve SLAM map quality

A thesis submitted to The University of Sheffield for the degree of Doctor of Philosophy

Christina Nefeli Georgiou

Department of Automatic Control and Systems Engineering

April 2017

Acknowledgements

Doing a PhD proved to be very different to what I originally anticipated and I am very grateful to all the people who have helped make it an interesting and enjoyable experience. First of all I have been very fortunate to have two very supportive supervisors, Prof. Tony Dodd and Dr. Sean Anderson, whom I would like to thank for all their comments throughout the years and all our technical and philosophical discussions on SLAM and probabilistic robotics. Tony's prompt and constructive feedback and Sean's attention to detail have always resulted in better work and more polished papers.

I would also like to thank my family for all their support throughout my PhD years, for setting the bar high and always believing in my ability to make it. Finally, to all my friends from Sheffield, Cambridge and Greece a big thank you for bearing with me through the difficult days and celebrating the good ones, this journey would not have been the same without all the experiences we have shared and all you have taught me.

Image acknowledgments

I would like to thank Jon Lipsky, lead developer of Elevenworks LLC, for allowing me to use a floor plan featured in the Elevenwork website and the University of West Florida for allowing me to use an image of a floor plan of the Heritage Halls of residence to test the image processing and mapping algorithms presented in this thesis.

Abstract

The problem of Simultaneous Localisation And Mapping (SLAM) has been widely researched and has been of particular interest in recent years, with robots and self-driving cars becoming ubiquitous. SLAM solutions to date have aimed to produce faster, more robust solutions that yield consistent maps by improving the filtering algorithms used, introducing better sensors, more efficient map representations or improved motion estimates.

Whilst performing well in simplified scenarios, many of these solutions perform poorly in challenging real life scenarios. It is therefore important to produce SLAM solutions that can perform well even when using limited computational resources and performing a quick exploration for time critical operations such as Urban Search And Rescue missions.

In order to address this problem this thesis proposes the construction of informative Bayesian priors to improve performance without adding to the computational complexity of the SLAM algorithm. Indoors occupancy grid SLAM is used as a case study to demonstrate this concept and architectural drawings are used as a source of prior information. The use of prior information to improve the performance of robotics systems has been successful in applications such as visual odometry, self-driving car navigation and object recognition. However, none of these solutions leverage prior information to construct Bayesian priors that can be used in recursive map estimation.

This thesis addresses this problem and proposes a novel method to process architectural drawings and floor plans to extract structural information. A study is then conducted to identify optimal prior values of occupancy to assign to extracted walls and empty space. A novel approach is proposed to assess the quality of maps produced using different priors and a multi-objective optimisation is used to identify Pareto optimal values. The proposed informative priors are found to perform better than the commonly used non-informative prior, yielding an increase of over 20% in the F_2 metric, without adding to the computational complexity of the SLAM algorithm.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robots operating in indoors environments such as hospitals, homes and rescue missions are becoming more common, leading to a growing demand for accurate robot navigation, localisation and mapping within known or unknown environments. A robot exploring an environment needs to have access to a map to be able to localise itself and keep track of its position. Moreover, in order to be able to map an unknown environment a robot needs to be able to accurately determine its pose at each time instant. If there is only partial or no knowledge of the area to be explored the robot needs to be able to localise itself and map its environment simultaneously. This problem, commonly referred to as Simultaneous Localisation And Mapping (SLAM), has been addressed in numerous ways over the years [47, 50]. The majority of commonly used solutions employ recursive Bayesian estimation to iteratively produce map and robot pose estimates. There is a vast literature of different SLAM approaches producing solutions improving different aspects of SLAM such as map consistency, map representation and sensors or filtering methods [53, 73, 132, 135].

Theoretical solutions to the SLAM problem often require infinite computational resources or linear robot dynamics, which are unrealistic requirements for real world applications. Simplifications such as using finite sets of samples to represent distributions [18] or linearising non-linear robot dynamics [82] are thus required to make such theoretical solutions applicable to real life problems. SLAM solutions proposed to date have managed to reduce computational complexity through the use of more efficient representations and to increase map and localisation accuracy [18, 21, 82]. However, since robots running SLAM algorithms need to be able to perform reliably in real life challenging scenarios, a focus on improving metrics without taking into consideration how such improvements translate to advantages in real world implementations is not sufficient. Whilst SLAM concepts

have been theoretically tested and perform well under controlled conditions and assumptions, real world implementations often do not meet such requirements, leading to poor performance in real life scenarios [111–113].

An application that has been very challenging to date for robots running SLAM is Urban Search And Rescue (USAR). Robots tested in real life rescue missions have performed poorly compared to their performance in simplified environments, failing to provide significant assistance to rescue personnel [13, 64, 83, 88, 111, 112, 114]. In order to be valuable in such scenarios robots need to be able to perform well even if computational resources are limited and sensor readings are unreliable, for example due to smoke interfering with laser range scanners. As robotics solutions become more and more common there is a need to produce systems that leverage all available information to perform well without requiring a number of environmental conditions to hold true. It is also important to design systems that do not require great increases in computational complexity to yield accurate pose and map estimates. This thesis builds on this premise to produce an improvement to current SLAM methods that yields better performance without increasing computational complexity or requiring any environment simplifications. It proposes the use of available prior information to construct informative Bayesian priors for SLAM, incurring only a one-off computational cost to construct such priors.

1.1 Motivation

There are a number of demanding real life applications where reliable SLAM solutions are necessary to ensure effective robot operation. Safety critical missions present a greater challenge, with robots operating in environments that may be more difficult to traverse, map, navigate and explore. The scenario that motivates the research presented in this thesis is USAR but the applications of this research are not limited to search and rescue. We use this environment as an example of a very challenging real life scenario in which producing more accurate maps without adding computational costs can be very beneficial.

The USAR environment is difficult and dangerous for human rescuers to explore and navigate [58, 76]. Rubble, collapse, smoke and untraversable areas are only a few of the potential difficulties rescuers face while operating in such an environment. In order to make such missions safer and reduce the risks to first responders the robotics community has proposed the deployment of robots to explore and map post-disaster areas and help locate and extract survivors [38, 106, 107, 113, 130, 142, 150].

Whilst progress has been made in the area of USAR robots, with many advances in terms of mechanical design, sensors and algorithms used for navigation, localisation and mapping, there are many remaining challenges [13, 64, 88, 112, 114]. One of the key challenges that is of interest in this thesis is accurate mapping of the disaster area, which can be particularly valuable in the case of buildings that have partially collapsed as a result of an earthquake, flood, mudslide or an explosion. In such a scenario producing an accurate map of the post-disaster area can help first responders plan safer and more efficient rescue missions by avoiding collapsed sections and planning safe, short routes to extract survivors.

So far, the performance of robots running SLAM algorithms in real life USAR missions has been disappointing, failing to provide valuable help to first responders [111–113]. This poor performance in real life environments can be attributed to unrealistic simplifications such as assuming accurate and reliable sensor data [53], using linearisations to simplify complex, non-linear robot dynamics [82] or assuming ample computational resources to achieve an accurate solution [135].

In order to be useful in a rescue mission SLAM systems need to meet the following criteria:

- Produce a usable map even if exploration time is limited for time-critical missions
- Produce a useful map even in scenarios where sensor readings are not reliable (laser range scanners can be unreliable if there is smoke in the environment, for example)
- Produce a useful map even if less accurate sensors are used
- Produce a useful map even if limited computational resources are available

Focusing on these proposed criteria will lead to solutions that aim to perform well in real life challenging applications and prioritise functional over mathematically elegant solutions.

1.2 The role of priors in SLAM

Bayesian methods dominate the estimation algorithms used in Simultaneous Localisation and Mapping (SLAM) [50], yet there has been very little research to date into how to effectively construct the Bayesian map prior using available information. This is the case even though the use of priors is a key distinguishing feature of Bayesian estimation, with the prior used in the majority of occupancy-based SLAM solutions being non-informative, assuming each location in the map

is equally likely to be empty or occupied. The lack of attention given to the map prior can be contrasted to the enormous wealth of raw prior information that is readily available and untapped for most mapping and localisation scenarios, such as architectural drawings for buildings, city maps for the urban environment, geographical surveys for the wider outdoor environment and even pipe network maps for the underground environment. There exists an important and unsolved problem, therefore, in how to optimally synthesize raw prior information into a form that is suitable for robot navigation.

A number of robotics applications have benefited from the use of available prior information to improve performance in real life situations. Prior information is used to improve navigation, localisation and mapping of self-driving cars or to improve the performance of odometry predictions [12, 24, 63, 101]. Prior information can also be used to define a known set of landmarks or landmark locations in SLAM [123, 124]. Unlike these methods which use raw prior information to impose constraints, this thesis proposes a method to process raw prior information and construct an informative prior map that can be updated using a recursive Bayesian mapping method.

Using prior information is very valuable to ensure systems perform well in real life, safety critical applications such as self-driving cars, where robust performance is favoured over generality or mathematical elegance. A similar approach is proposed in this thesis for robots running SLAM algorithms in safety critical applications such as USAR. A non-informative prior map is commonly chosen in an attempt to provide a solution that does not depend on having access to prior information. However, for most real life situations, sources of prior information are available. Refraining from using prior information simply to produce a more general solution does not result in a system that can perform well in challenging real life scenarios. Conversely, incorporating prior information can improve performance and provide a more robust system in case of sensor failure, for example.

Given the Bayesian formulation of the majority of state-of-the-art SLAM solutions informative Bayesian priors can be used to yield a number of advantages:

- In recursive map estimation they can produce more accurate maps if a time constrained exploration is performed
- They can provide information about unexplored areas
- They can help provide information about the environment even in case of sensor failure or malfunction
- They do not add to the cost of running the SLAM algorithm itself and can

be incorporated to any SLAM implementation that uses recursive Bayesian mapping

This thesis uses indoors occupancy grid SLAM as a case study to explore this concept but the methods used could also be extended to an outdoors scenario.

1.3 Research aims and scope

The main aim of this thesis is to propose a method to improve SLAM performance in challenging real life scenarios in which current state-of-the-art solutions struggle. The motivating example has been USAR due to the great number of challenges it presents in terms of sensor reading accuracy, available computational resources and exploration time but also the great potential to save lives if robotics solutions perform as required. Given the potential advantages of using SLAM priors we aim to produce informative priors that can improve map accuracy in such environments. In order to be useful to first responders the map produced needs to be accurate in terms of representing the environment in a way that is legible and easy to understand by non-expert human responders.

Unlike many SLAM approaches in the literature that provide feature-based maps to be used by robots to localise or plan routes [40, 74, 77, 116], our focus is not localisation but mapping, and our end user a human and not a robot. Given this definition the map representation of choice is occupancy grid maps that resemble floor plans and are easier for rescuers to understand. Even when using an occupancy grid map representation, however, modern systems often produce a final map that is not always easy to interpret by a non-expert. Although many of the maps produced using state-of-the-art occupancy grid SLAM methods yield maps that look impressive to non-experts at first sight, very few of them can be used reliably to understand an indoors environment. Unexplained artifacts, lines at an angle or missing segments can correspond to mapping errors or, in the case of a USAR operation, to collapsed areas.

Mapping errors are more common in a USAR environment. Dust can interfere with range sensors, poor light can make visual feature detection impossible. Mapping errors caused by such phenomena might not be important in some applications but in a safety critical mission it is imperative to convey accurate information even if sensors perform unreliably. Another problem faced in time-critical missions is limited exploration time. An incomplete exploration can lead to an unintelligible map that effectively provides no information to first responders. Moreover, computational resources may be limited due to limited onboard pro-

cessing power and insufficient bandwidth in communication channels may limit the amount of information that can be sent to a remote computer for processing.

The above challenges highlight the need for a SLAM solution tailored to this type of challenging scenario. Given this scope, our research aim is to use informative priors to produce a SLAM solution which, compared to the same solution using a non-informative prior

- Yields a more accurate map: an accurate map is defined as one that agrees with the ground truth, correctly detecting wall sections
- Provides information about unexplored areas: information about walls in unexplored areas needs to be provided
- Yields a more accurate map throughout the exploration: accuracy needs to be reached and maintained throughout the exploration
- Maintains improved accuracy when using a small number of sensor readings: accuracy is improved even if few sensor readings are used
- Computational cost associated with such improvements is minimised: improved accuracy should not incur great computational cost increases

Using these aims, we define the following goals for this scenario. In terms of the final map produced, we aim to improve accuracy compared to that produced using a non-informative prior in four ways

- **Overall accuracy**: the map needs to correctly represent the environment. This consists of two aspects
  - Detecting as many of the walls as possible
  - Ensuring that detected occupied space corresponds to true occupied space
- **Accuracy of unexplored areas**: this is the same as the accuracy goal but applies to unexplored areas; this will be reflected in the overall accuracy but is also worth examining separately
- **Accuracy vs number of scans**: accuracy needs to be achieved even for a smaller number of scans
- **Accuracy vs exploration time**: map accuracy needs to reach and maintain a high value as early as possible during exploration

A detailed explanation of performance metrics that measure the extent to which these goals are met is given in Chapters 4 (accuracy measured using precision and recall) and 5 (accuracy measured using the evolution of the F_2 metric with time and for different numbers of scans). Improved accuracy as defined above is always relative to that achieved using a non-informative prior. The maximum achievable accuracy in a given scenario will depend on factors such as sensor accuracy, grid resolution and chosen SLAM algorithm implementation and as such no absolute accuracy value goals are defined.

1.4 Proposed approach

In order to construct an informative Bayesian prior map a source of prior information needs to be identified and the information needs to be processed to be placed in a prior format. Road network maps or aerial photographs can be used for outdoors applications and architectural drawings and floor plans can be used for indoors applications.

Indoors mapping is used as a case study to demonstrate this concept and architectural drawings and floor plans are used as a source of prior information. An occupancy grid FastSLAM implementation [109] is chosen to allow for the study of recursive mapping without the need to address localisation. Occupancy grids represent the area to be explored by a grid of cells each associated with a probability of occupancy. They are the map representation chosen in this thesis because the incorporation and visualisation of prior information is relatively straightforward. The map at time $k = 0$ is the Bayesian prior constructed based on available information and if no information is available a priori a non-informative Bayesian prior is used.

The main aim of this thesis is to propose a method to construct optimised informative indoors priors. Therefore the main focus is an algorithm to process the drawings, extract relevant information and convert it into an appropriate prior format through the optimisation of prior parameters.

The benefits of using such priors are tested through a mapping implementation that assumes robot poses are accurate and known a priori. This assumption allows for a more comprehensive study of possible prior parameters and their effects. Given the separation of localisation and mapping in FastSLAM the results obtained in this thesis are also applicable to SLAM systems in which poses are not known and are estimated using methods such as Monte Carlo localisation. An occupancy grid FastSLAM implementation is used to test the proposed priors in experiments using a real robot to experimentally validate simulation results.

1.5 Contributions

The main contribution of this thesis is a method to construct a Bayesian prior for recursive probabilistic mapping using available prior information. The contributions of this thesis are as follows:

- An analysis of the role of priors in occupancy grid SLAM and desirable prior properties.
- Given the shortcomings of existing architectural and floor plan processing methods a novel method for door and wall detection in architectural drawings and floor plans is proposed in Chapter 3. This method is tailored to the goal of correctly extracting wall locations and is found to outperform the commonly used dilation method to determine wall locations. The proposed method is found to yield good qualitative and quantitative performance across all drawings, unlike the dilation method which does not generalise well for different types of drawings.
- Due to this focus on wall detection, existing methods used to assess the quality of information extracted from architectural drawings and floor plans are found to be ill-suited to this application. A novel approach to assess the quality of extracted structural information from an architectural drawing is proposed in Chapter 3, viewing the assessment as a binary classification problem, aiming to correctly classify walls and empty space.
- The problem of assessing map quality is challenging and there is no universally accepted method to do so, with the majority of proposed methods being tailored to the application or the SLAM algorithm used. Given this limitation a novel approach is proposed in Chapter 3 to assess the quality of occupancy grid maps. The problem of robot mapping is formulated as a binary classification of all cells of an occupancy grid as empty or occupied space.
- The extracted structural information then needs to be placed in a suitable format in order to construct a prior map. A method to convert structural information extracted from an architectural drawing to an occupancy grid prior is proposed in Chapter 4 and design variables to be determined are identified.
- Due to the novelty of the proposed approach there have been no studies to assess how prior values assigned to structural information extracted from an

architectural drawing or floor plan affect the quality of the map. Chapter 4 addresses this gap in the literature and conducts a study of how prior values of occupancy assigned to detected walls and empty space affect map quality.

- The development of an overall algorithm to construct optimised Bayesian priors using an architectural drawing or floor plan is presented in Chapter 5, proposing prior values that yield good qualitative performance across all drawings. Using such priors results in more accurate maps without adding to the computational cost of running the SLAM algorithm.

- A qualitative comparison of maps produced using a non-informative prior and those produced using the proposed informative prior is conducted in Chapter 5 in order to explore the potential benefits of using an informative prior. The proposed method is found to outperform the commonly used non-informative prior and yield qualitatively good maps even if a coarse grid is used or an incomplete exploration is performed.

- A quantitative benchmarking of maps produced using a non-informative and the proposed informative prior is conducted in Chapter 5 to quantify the benefits of using the proposed approach. Maps produced using the proposed informative prior are found to outperform those produced using a non-informative prior.

- An experimental validation of using the proposed prior in SLAM is presented in Chapter 6, with results showing improved performance over using the non-informative prior. The proposed prior is found to perform better both in a qualitative and in a quantitative sense and to provide information about unexplored areas that makes it easier for human users to interpret the map.

1.6 Thesis overview

Chapter 2 presents an overview of the USAR environment and highlights the advantages of using robots to map and explore post-disaster areas and assist with rescue missions. The remaining challenges such an environment presents to robotics solutions are presented and indoors SLAM is identified as the focus of this thesis. An overview of SLAM solutions used to date as well as state-of-the-art solutions are presented. A gap in the literature is identified in the area of constructing optimised informative SLAM priors to improve performance and the significance of priors in recursive Bayesian SLAM is highlighted.

Chapter 3 explores architectural drawings and floor plans as a source of prior information for indoors SLAM. A novel method to process such drawings and extract structural information is proposed, as well as a method to quantitatively assess the quality of the extracted information. The problem is presented as a binary classification of all pixels in the drawing, with each pixel belonging to either walls or empty space. Recall, precision and false positive rate can then be used as performance metrics. Results show this method yields a precision of at least 98%, recall of at least 61% and a false positive lower than 0.09 % for all drawings tested. Results are also benchmarked against those obtained using dilation to detect walls and the proposed approach is found to generalise better, yielding good performance across all drawings.

Chapter 4 conducts a study of possible prior values of occupancy to assign to detected walls and empty space in an architectural drawing. An occupancy grid mapping simulator for Matlab is proposed and presented in order to test the benefits of using different prior values. Possible combinations of prior values to assign to detected walls and empty space are then tested using this simulator. A novel approach to assess map quality is proposed, viewing the assessment as a binary classification of all map pixels as being walls or empty space. A precision-recall analysis is then used to assess performance. The objectives of optimising precision and recall are found to be conflicting and multi-objective optimisation is used to find Pareto optimal prior values. Shortlisted Pareto optimal values yield significantly improved qualitative results compared to values that optimise a single objective. Finally, the Pareto optimal shortlisted values are tested to determine values that yield good qualitative performance across all drawings. A k-means clustering algorithm is used to identify clusters within the prior value solutions and a cluster containing prior values that yield good qualitative performance for all drawings is identified. Values within that cluster are then tested to identify the prior values that yield the best qualitative performance.

Chapter 5 tests the benefits of using an informative prior, both in a qualitative and quantitative sense and the effects the quality of the extracted prior information has on the final map are examined. The proposed informative priors yield improved performance compared to maps produced using a non-informative prior, resulting in an increase of over 20% in the F_2 metric. Using informative priors is also found to outperform using non-informative priors where a computationally efficient solution is required, yielding an accurate map even if an incomplete exploration is performed or the sensors used become unreliable.

Chapter 6 presents the occupancy grid FastSLAM implementation used to obtain experimental results. An overview of sensor and motion models used is given.

Larger scale experimental results obtained using a turtlebot mounted with a Kinect sensor are presented. The results obtained using the proposed prior are compared to those obtained using a non-informative prior and the proposed prior is found to perform better, further confirming the results obtained in Chapter 5. Experiments are also conducted to test how localisation accuracy is affected by the quality of the prior map. A drawing as-is is compared to using the extracted prior. Localisation quality using the drawing as-is is found to be poorer, and significantly worse as exploration progresses, confirming the advantages of extracting walls to create a prior.

Chapter 2

Robot mapping and localisation for USAR

The use of robots in USAR missions has been of great interest in recent years due to disasters such as the terrorist attack of 9/11, the Katrina hurricane, the earthquakes in Nepal and the Fukushima-related disasters in Japan [111–113]. In such scenarios the risks to first responders can be minimised by deploying a robot or team of robots to explore and map the post-disaster area and locate and even extract survivors. Despite their potential to be very valuable, robots deployed in real life rescue missions have been for the most part disappointing [13, 64, 83, 88, 111, 112, 114]. Using a robot to produce accurate maps quickly and efficiently remains an unsolved problem, with most state-of-the-art solutions performing well in simulated or greatly simplified environments but struggling to meet the speed, reliability and accuracy requirements of real life missions. This chapter presents existing solutions, highlights their shortcomings and identifies gaps in the literature that make current solutions insufficiently accurate and efficient for rescue missions.

This chapter is structured as follows. Section 2.1 gives an overview of USAR operations and the value robots can add to rescue missions, identifying remaining challenges for robot systems used in USAR. The area of SLAM is identified as a particularly significant challenge and an overview of SLAM solutions found in the literature is given in Section 2.2. Advances and established methods in the areas of different sensors, map representations and filtering algorithms as well as state-of-the-art solutions are identified. Finally the effect of the Bayesian priors used in SLAM is discussed and their role in occupancy grid SLAM, the focus of this thesis, is presented in Section 2.3.

2.1 Robotics solutions for USAR

Robots are commonly used in tasks that are too dull, dirty, difficult or dangerous for humans to perform. USAR operations are an example of such a task since they take place in environments that are very challenging for rescuers to explore and navigate. Rubble and potential collapse, smoke and fire, chemical, biological or nuclear contamination make these operations extremely dangerous to first responders who need to venture into post-disaster areas without knowing what to expect. Deploying a robot equipped with sensors to explore the area and produce a map for first responders to use, or assist with rescue missions is therefore a solution that has been proposed by the robotics community to help make rescue missions safer and more efficient [38, 106, 113, 142, 150].

2.1.1 The USAR environment and operations

USAR [58] operations have received a lot of media attention in recent years due to disasters such as the terrorist attack at the World Trade Centre, the Katrina hurricane, the earthquakes in Nepal and the Fukushima-related disasters in Japan. Responding efficiently and effectively after such disasters occur requires complex rescue operations that minimise risks to first responders [76]. The post-disaster areas present a number of difficulties, dangers and hazards and operations need to be carefully planned to ensure efficient, quick and safe rescue of survivors. Given the safety-critical nature of these operations, improving their efficiency and reducing risks to the lives of first responders has been of great interest, with many researchers proposing robotics solutions to assist with such missions [38, 106, 107, 113, 130, 142, 150].

The environment

The USAR environment is very difficult and often dangerous for rescuers to explore and navigate [58]. Partial or total building collapse can result in structurally unsafe partially collapsed areas or piles of rubble that are difficult to explore and access. An example of such a disaster area after the 2015 earthquakes in Nepal is shown in Figure 2.1, which highlights the varying levels of possible building collapse. Fully collapsed sections are difficult to explore and partially collapsed buildings can be unsafe to enter and can contain smoke, dust and structurally unstable sections. Another common disaster is fire in buildings, where the smoke and heat can disorient rescuers and cause breathing problems or injure them. Immediate response is required to rescue victims that may be injured or unconscious

Figure 2.1: Partially and fully collapsed sections of buildings after the 2015 Nepal earthquake. Fully collapsed areas are very difficult to explore and partially collapsed areas can be structurally unstable and thus unsafe (image reproduced from [49] under the Creative Commons license).

[144]. Finally, buildings that have suffered chemical, nuclear or biological contamination may need to be explored [113]. Such buildings present great dangers to the health of responders and no human teams can be safely deployed to explore them.

The operation

The guidelines for rescue operations vary between countries and thus there is no universally adopted search and rescue procedure. Rescue handbooks such as [133] for Australia and [102] for the UK describe the structure of such operations. According to a study of Australian rescue operations [64], the main aim is to locate and rescue victims as fast as possible and without excessive risk to the lives of rescue personnel. The operation also needs to be efficient, minimising the setup time required. Time is more critical in situations where the disaster area changes very rapidly such as fire [144] and less critical in operations where the scene is less likely to change significantly such as building collapse.

The need for robots

Search and rescue environments present numerous hazards and difficulties to human responders. Difficult or no access, potential building collapse, extreme heat, radiation or poisonous gases are only a few of the difficulties rescue personnel may face. Rescue operations can thus greatly benefit from integrating robotics solutions that help assess damage, map and record video of areas that are inaccessible or dangerous to humans or trained canines. In recent years robots designed and/or programmed by various universities have been deployed in rescue operations such as the World Trade Centre [111], the mudslides in La Conchita, California in 2005 [112] and the Fukushima contaminated areas [113].

2.1.2 Progress achieved and remaining challenges

Robotics research for USAR has had a few successes in recent years. Robot solutions that can navigate challenging terrain have been designed [38], with some using more suitable tracks or wheels [113], others using innovative designs such as a snake-like structure to navigate areas that are difficult to access [154] and some using aerial vehicles [26] or even teams of different robots [107, 130]. Robot localisation and mapping is a topic that has been the focus of much recent research and advances are being made in terms of map quality [106, 142, 150], localisation accuracy [37, 54] and system scalability and robustness [98].

However there are many remaining challenges which can be grouped into the following categories [13, 64, 83, 88, 111, 112, 114]:

- **Robot design (mechanical)**: different robot designs to tackle challenging terrain and hazardous environments
- **Sensors**: more durable/suitable sensors that provide more accurate data for mapping, localisation and exploration
- **Communications**: more reliable data transmission
- **Autonomy/user interface**: achieving higher levels of autonomy and making the proposed system user friendly/easy to use by non-experts and rescuers
- **Robot localisation and mapping**: different ways to localise robots within the environment and produce a map of the post-disaster area

The focus of this thesis is accurate mapping and localisation, particularly in challenging real life scenarios such as USAR and partially collapsed buildings. A robot exploring a partially collapsed building can produce a map to be used

by rescue personnel to plan more efficient and safer missions, access casualties and survivors and minise risks to first responders. Accuracy is important because mapping errors may be misinterpreted as structural damage and mislead rescuers.

2.1.3 Robot localisation and mapping

Attempts have been made to produce maps of disaster areas but most of them have struggled with real life challenging environments. After the World Trade Centre terrorist attack roving robots were used to search for victims, paths through rubble that could be quicker to excavate and hazardous materials and perform structural inspection [88, 111]. The inside of the buildings was dark and covered with thick dust and mud which caused problems with navigation [88, 111] and the post-disaster area lacked easily distinguishable landmarks or recognisable locations which made mapping using cameras difficult.

Localisation and mapping improvements proposed to date aim to produce faster, more robust solutions that yield consistent maps. This focus on improved performance has lead to solutions that perform well in simulated or simplified environments but struggle in complex real life environments such as USAR [13, 64, 83, 88, 93, 111, 114, 130]. Accurate, real time SLAM for indoors USAR scenarios such as partially collapsed buildings remains an unsolved problem. In order to be useful in USAR, SLAM-produced maps need to be more accurate and account for any out-of-the ordinary observations whilst also ensuring computational efficiency.

The localisation and mapping requirements for USAR applications can be summarised as [64, 88, 111, 112, 147]:

- High computational efficiency
- High accuracy and reliability
- High accuracy for time-limited/incomplete exploration
- Good readability of map by rescue personnel

It is therefore important to produce more SLAM solutions that focus on good performance with limited resources in challenging environments in order to make robots useful contributors rather than cumbersome observers to USAR missions. The SLAM problem [50] and advances made as well as their relevance to USAR are discussed in the next section.

2.2 Simultaneous Localisation And Mapping

Robots used in real life environments such as USAR need to be able to map their environment and at the same time localise themselves within it. This section presents this problem commonly referred to as Simultaneous Localisation And Mapping (SLAM) and lists commonly used solutions as well as remaining challenges.

2.2.1 Problem statement

SLAM is the problem of placing a robot in an environment and having it produce a map of the area and keep track of its location within that environment simultaneously [50]. In order to achieve this task the robot needs to have access to a map to be able to determine its location but it also needs to be able to track its location within the environment at each time instant to be able to map its surroundings correctly.

The structure of the SLAM problem, Figure 2.2, can be broken down into the following elements. Sensors are used to detect objects to produce a map of the area and a motion model or odometer is used to predict the robot's pose. However, sensor data are not always accurate and they cannot always be interpreted correctly and motion models are imperfect and cannot predict errors due to phenomena such as wheel slipping. In order to address these problems a SLAM filtering algorithm based on probabilistic estimation is used, and the noisy robot pose and map estimates are updated recursively. In the Bayesian formulation of SLAM a prior can also be used to incorporate any information about the environment that is available a priori.

As shown in Figure 2.2 the problem of robot SLAM can be broken down to a number of sub-problems, namely [50]:

- A choice of sensors
- A choice of a map representation
- A choice of a motion model for the robot
- A choice of a filtering algorithm that combines pose and map data to solve the SLAM problem

2.2.2 Environment representation and robot sensors

The robot environment is the indoors or outdoors space that the robot navigates or operates within. Data collected by sensors mounted on the robot can be used

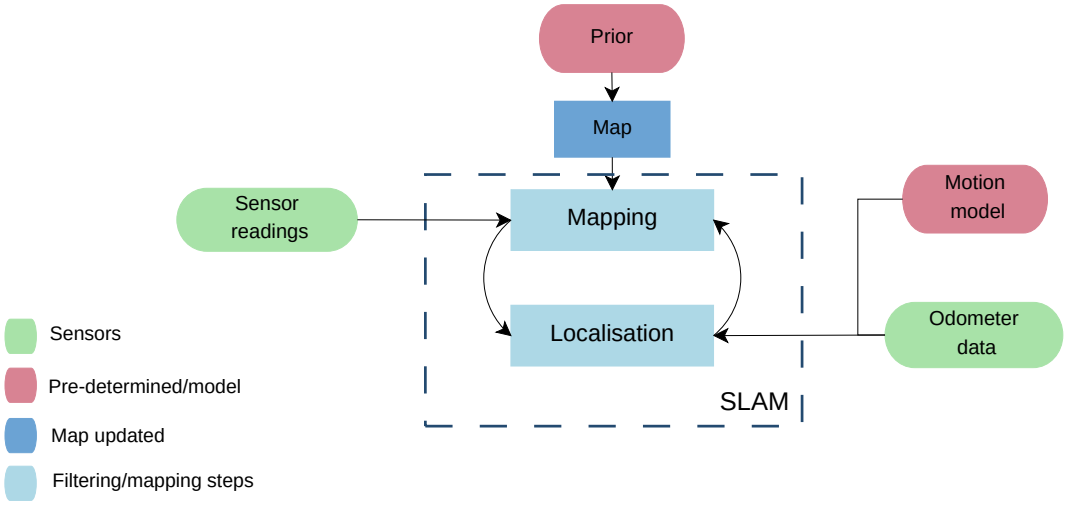


Figure 2.2: The structure of the SLAM problem: sensor data, prior information and motion model predictions are input to the SLAM algorithm that recursively maps the environment and localises the robot within it.

to construct and update a map of this environment. Depending on the sensors the robot uses there are a number of possible map representations that are commonly used in the literature [19, 29].

An overview of possible map representations can be seen in Figure 2.3. These can be grouped as follows:

- **Metric maps** can be feature-based or occupancy-based
  - **Feature-based maps** contain the location of a number of unique and/or easily distinguishable features
  - **Occupancy maps** look more like floor plans of the explored area, identifying occupied and unoccupied space in the environment
- **Topological maps** give an overview of areas of interest and their relative positions without giving any metric information

Most of these representations are used to produce 2D maps but can also be extended to 3D representations.

Feature-based maps

This type of map represents the environment as a collection of locations of selected features or landmarks. An example of such a map can be seen in Figure 2.4. Cameras are commonly used to detect uniquely identifiable visual features within the environment using feature detectors such as Random Sample

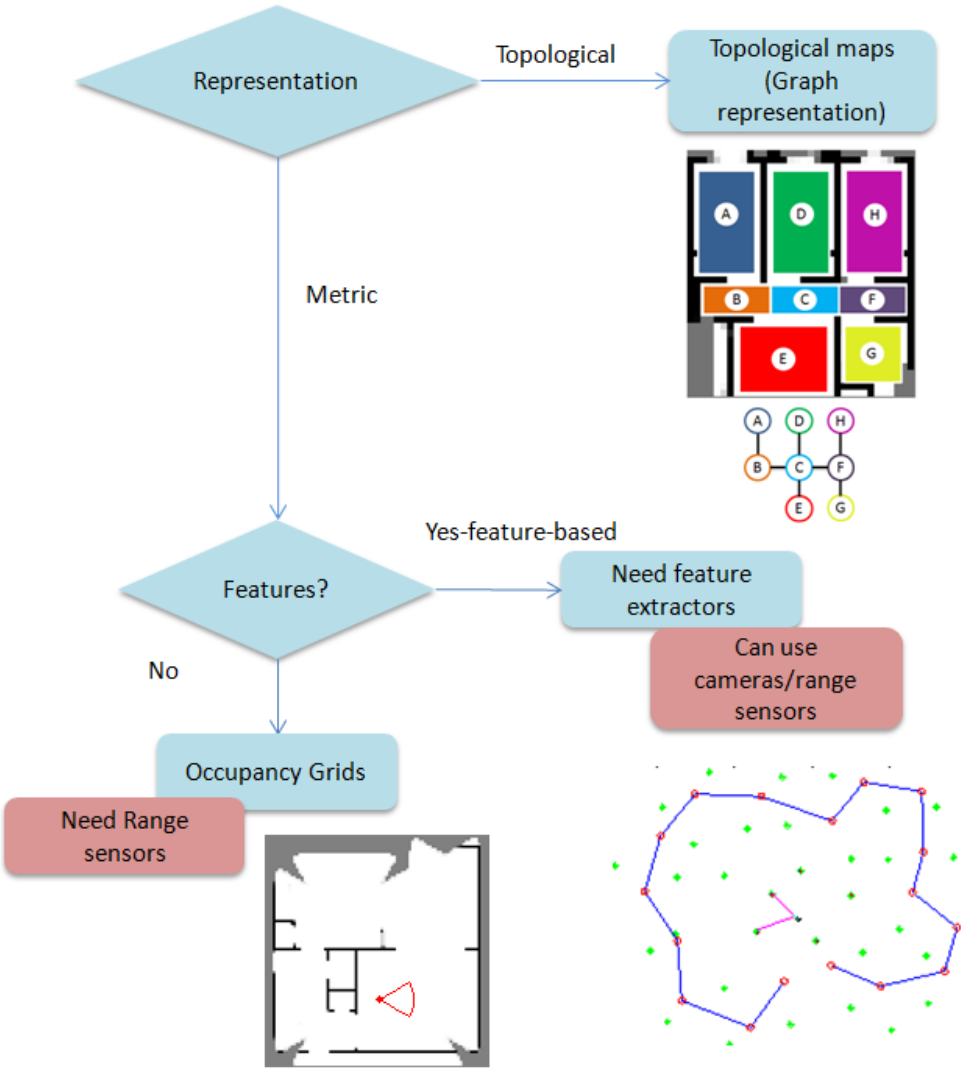


Figure 2.3: Different map representations: Depending on the choice of map representation different sensors and SLAM algorithms may be used (landmark map image was produced using the FastSLAM Matlab code found in [141]).

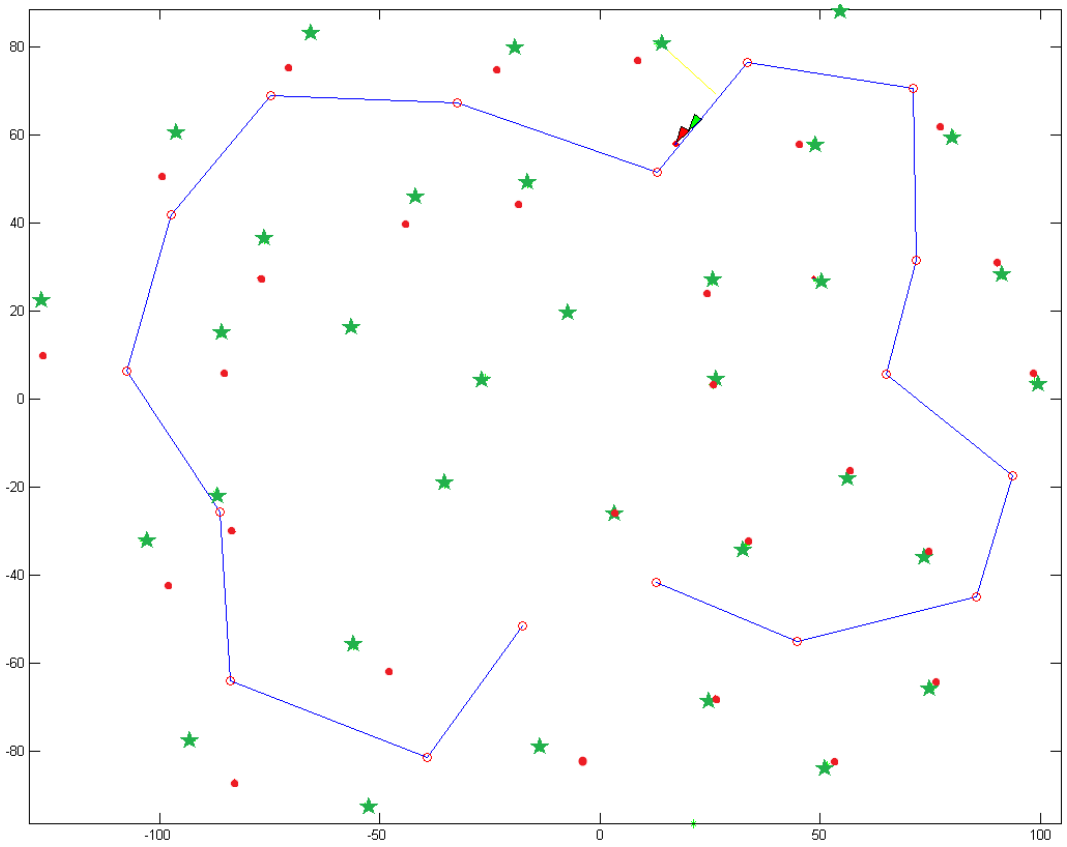


Figure 2.4: Example of a feature-based map: the true robot trajectory is shown in blue, true landmark positions are drawn as green stars, predicted landmark positions as red dots, the true robot position as the green arrow and the predicted position as the red arrow. This figure was produced using the FastSLAM Matlab code found in [141]. Cameras can be used to identify features and estimate their distance from the robot.

Consensus (RANSAC) [55]. There are many SLAM implementations using vision and visual feature extraction [40, 41, 74, 77, 116] and an overview of visual SLAM methods can be found in [95]. Distance sensors can also be used to detect features such as walls by searching for linear sections in the environment, for example [96, 122, 148]. Using distance sensors to detect walls as features is less common, however, and most feature-based systems tend to use cameras.

The main advantage of this type of representation is its sparsity, since only a limited number of features and their locations need to be maintained. However, selecting a sufficient number of uniquely identifiable features and being able to correctly detect them using on-board cameras is a difficult problem. Selecting suitable features is non-trivial since many environments such as offices do not always contain a sufficient number of uniquely identifiable landmarks. Even if there are

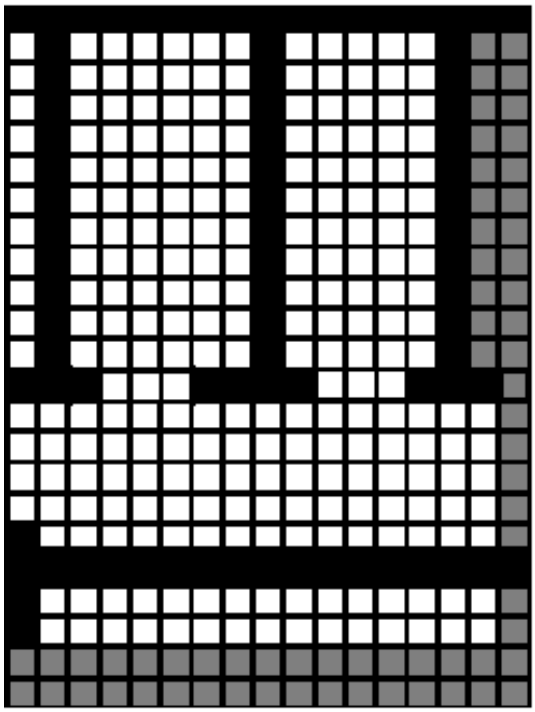


Figure 2.5: Example of an occupancy map: the environment is split up into square cells and a probability of occupancy is updated for each; white squares have a high probability of being empty, black squares a low probability of being empty and grey squares are equally likely to be occupied or unoccupied.

enough unique landmarks within the environment the problem of robustly and reliably identifying them, often called the data association problem [20], remains unsolved. Feature-based maps are often chosen for the exploration and mapping of outdoors environments since their sparsity is advantageous when mapping a very large area [97, 104].

Occupancy maps

Occupancy grid maps aim to distinguish between empty and occupied space within the environment [51]. In order to do so the area to be explored is split into a grid of squares as shown in Figure 2.5. Each square is assigned a prior probability of occupancy, typically 0.5. Assigning such a prior probability indicates that there is no available information about the environment a priori since each square is assigned a prior probability of 0.5 signifying it is equally likely to be occupied or unoccupied. As new sensor data is collected the occupancy value of each square is recursively updated to incorporate this data.

This type of representation is most commonly used to represent data collected

by robots mounted with distance sensors such as ultrasound or laser range finders and many implementations of SLAM using this approach have been proposed [53, 63, 67, 69, 106, 121, 152]. An example of an occupancy grid map can be seen in Figure 2.5, where white areas have a high probability of being unoccupied, black areas a low probability of being unoccupied and grey areas have not been explored (so are equally likely to be unoccupied or occupied).

This representation has a number of advantages [108, 136]. Information is collected and represented for all map locations with an explicit representation of empty space. This representation also allows for different models of sensor uncertainty to be used when updating occupancy values [136]. For indoors environments this is an intuitive map representation for human responders to use to plan USAR missions, for example, since it greatly resembles a floor plan.

However, this representation has a number of limitations. Depending on the size of the area to be mapped and the chosen grid resolution it can be computationally costly, having to update and maintain occupancy values for a large number of grid cells. A finer grid results in better map quality, however, so there is a trade-off between map accuracy and computational complexity. DP SLAM aims to address this issue by using a tree representation to efficiently store map data but assumes very accurate sensors [53], an unrealistic assumption for real-life applications. Moreover, if the model for sensor uncertainty used to update occupancy values is incorrect the map produced could be inaccurate [136].

Topological maps

This type of map does not rely on metric measurements but rather represents the environment in terms of locations of interest and paths connecting these. Graphs are commonly used to represent such maps as shown in Figure 2.6, with each node corresponding to a different room and each edge showing what other rooms are accessible from that node. Information is collected using robot sensors and the most likely graph configuration given sensor readings is calculated offline after all data are collected [34, 56].

The advantage of such a map is that it gives an overview of locations and traversable paths and can be used for route planning and optimisation. The commonly used graph representation also allows the use of standard graph optimisation techniques. This type of map can be useful in terms of giving an overview of the structure of large indoors environments. However, this method cannot be used to produce a map while the robot is exploring since all data needs to be collected before the map can be constructed. It also provides no information about the structure of each node (or room in the case of the map in Figure 2.6). It does not

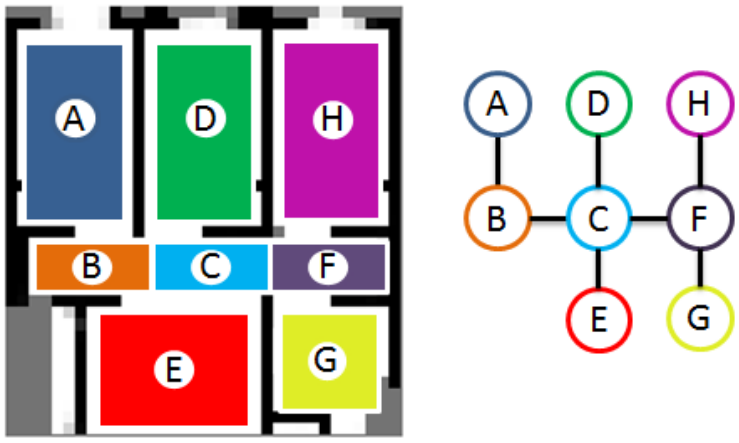


Figure 2.6: Example of a topological map: the environment is split up into areas A-H and the relevant graph can be drawn. Graph nodes represent different rooms in the environment and graph edges represent possible routes between different rooms. Weights can also be assigned to each edge to indicate how far rooms are from one another.

provide a full physical representation and is thus ill-suited to applications where a detailed mapping of an area is required such as USAR.

Hybrid maps

Each map representation has a number of advantages, with different representations being better suited to different applications. Some research has combined these methods to produce solutions that leverage the strengths of different methods.

A hybrid topological and metric map is proposed in [150] for use in indoors environments containing ruins, such as partially collapsed buildings. A topological map provides an overview of the building structure such as the number of rooms and accessibility of each room, an occupancy map is used to detect obstacles and features are used to identify different locations. The system has not been benchmarked against solutions using different map representations and so quantifiable advantages have not been identified. Another topological-metric hybrid solution [91] uses a graph to plan the overall mission and a metric occupancy map to localise within each node of the topological map.

A combination of feature and occupancy SLAM is proposed in [155] in which a robot switches between a feature-based and an occupancy-based map depending on whether it is navigating outdoors or indoors. A reinforcement learning approach is used to train the map type selection algorithm and therefore this ap-

proach can suffer from problems such as overfitting and performance is dependent on the training set of choice.

Whilst hybrid maps allow the use of the most appropriate representation for each task they also have an increased computational cost since a number of different representations need to be maintained and the difficult task of choosing an optimal representation for each location/task needs to be performed.

2D and 3D

All the map representations described above can be extended to 3D and producing 3D maps has been a topic of interest in the field. An array of laser range scanners [75, 79] or a rotating laser range scanner [151] can be used to provide 3D information. Depth cameras (time-of-flight or stereo vision) [73, 132] and the Microsoft Kinect sensor [72] can also be used to provide 3D data. 3D mapping can also be achieved using flying robots mounted with distance sensors [130]. These systems use representations such as object point clouds to provide 3D maps of the environment. 3D mapping is computationally more expensive and accurate 3D mapping remains an unsolved problem.

2.2.3 Filtering algorithms

Different types of filters can be used in SLAM to produce estimates of robot pose and landmark locations/occupancy maps. There are two important elements to this problem: the update of the robot pose and the update of the map/landmark locations [50].

In order to predict the robot's pose a model of robot dynamics can be used to take a control input and a starting pose for the robot and produce an estimate of the robot pose, $\mathbf{x}_k$, at the next time step. Alternatively odometry measurements can be used to predict the robot's pose $\mathbf{x}_k$ but, in order to simplify the explanation of different methods, we will assume that all predictions use a robot dynamics model. Both odometry predictions and model predictions are inaccurate: models cannot perfectly model all real-world effects and events such as wheel slipping can cause erroneous odometry readings.

Similarly, a model of what landmark/occupancy values are observed, based on the robot's pose and the set of landmarks/the map up to that point, can be used. A perfect mapping between sensor readings and landmarks in the map/map locations cannot be guaranteed due to feature detector inaccuracies.

Given the robot dynamics and observation model the aim is to produce an estimate from the noisy motion model and use that to update landmark locations/the

map. The map can then be used to improve pose estimates. Thus estimates of robot poses and landmarks/ the map as well as the uncertainty associated with each prediction can be produced recursively. There are a number of different filters that are commonly used in the literature to perform this estimation and some of the most popular choices are presented in the following sections.

Probabilistic formulation

SLAM performs a probabilistic estimation to determine the robot's pose and a map of the environment given sensor readings, control inputs and the robot's initial pose. Formally, it aims to compute the joint posterior of the robot's pose $\mathbf{x}_k$ and the map $\mathbf{m}$ [50]

$$p(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \tag{2.1}$$

for all times k, where $\mathbf{x}_k$ is the state vector describing the robot's pose, $\mathbf{m}$ is the map of the environment (or a selection of landmarks for feature-based SLAM), $\mathbf{Z}_{0:k}$ is the complete set of sensor observations, $\mathbf{U}_{0:k}$ the history of control inputs from time 0 to time k and $\mathbf{x}_0$ is the robot pose at time $k = 0$. The aim is to estimate the robot pose for all times k and the map of the environment given sensor readings, control inputs and a starting pose. SLAM is solved recursively, producing map and pose estimates at each time step.

The joint posterior in (2.1) can be written

$$p(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) = p(\mathbf{z}_k | \mathbf{x}_k, \mathbf{m}) \frac{p(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{p(\mathbf{z}_k | \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \tag{2.2}$$

with the step shown in Equation 2.2 commonly referred to as the measurement update that incorporates observation data $\mathbf{z}_k$ and

$$p(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \int p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k) \times p(\mathbf{x}_{k-1}, \mathbf{m} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) d\mathbf{x}_{k-1} \tag{2.3}$$

is the time update, incorporating new pose information.

In order to be able to perform these two updates $p(\mathbf{z}_k | \mathbf{x}_k, \mathbf{m})$ and $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k)$ need to be known. This however is not the case for real life operations and an estimator needs to be used to perform the measurement and time updates. There are two types of methods commonly used to perform this approximation. A parameterised model of $p(\mathbf{z}_k | \mathbf{x}_k, \mathbf{m})$ and $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k)$ can be used and Kalman or Information filter based methods fall into that category. However, such methods require linearisations to produce linear robot dynamics and assume Gaussian noise, but neither of these assumptions is realistic for real world systems. Another

set of solutions employ a particle filter to approximate these two distributions by a set of samples. This type of solution does not have any linearity or Gaussianity requirements but good performance depends on the number and quality of samples.

These two types of methods are explored in following sections, presenting Kalman, Information and Particle filters. The merits and drawbacks of each method are presented, highlighting remaining challenges.

Kalman filters

Kalman filters use a state-space model representation with additive Gaussian noise [50] to approximate Equation 2.2. They recursively process noisy data to produce estimates of the system state that minimise the error covariance.

The update for the pose estimate, $\mathbf{x}_k$, modeled as the sum of the robot dynamics and Gaussian noise, is defined as

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k \tag{2.4}$$

where $f()$ models the robot dynamics, $\mathbf{x}_k$ is the robot pose at time k, $\mathbf{u}_k$ the control input at time k and $\mathbf{w}_k$ additive, zero mean uncorrelated Gaussian motion disturbances with covariance $\mathbf{Q}_k$.

Similarly, observation values $\mathbf{z}_k$ are updated using an observation model and additive Gaussian noise

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{m}) + \mathbf{v}_k \tag{2.5}$$

where $\mathbf{z}_k$ are the sensor observations at time k, $h()$ is the observation model (what the observation is likely to be given the map/landmarks and robot positions), $\mathbf{x}_k$ the robot pose at time k, $\mathbf{m}$ the set of known landmarks or the map up to that time and $\mathbf{v}_k$ is additive, zero mean uncorrelated Gaussian observation noise with covariance $\mathbf{R}_k$ [50].

The standard Kalman filter assumes $f()$ and $h()$ are linear functions which is, however, not the case in most real-world robotics applications. In order to overcome this problem a Taylor series linearisation of the equations about the location of the detected landmarks can be performed, producing the Extended Kalman Filter solution (EKF). Performing this linearisation leads to inaccuracies, however [48], since the Taylor series only yields an approximation.

A number of studies of the consistency and convergence of EKF SLAM have been conducted [21, 82]. These indicate that estimates produced are very opti-

mistic once the true uncertainty in vehicle heading exceeds a limit. This failure is difficult to detect without having access to ground-truth data and commonly used solutions such as adding stabilising noise are not effective. An approach that uses Jacobians of the state and measurement models that are evaluated using the first estimates of all landmark positions and robot poses to improve consistency is proposed in [80] and is shown to yield a performance close to the ideal EKF based on simulation results.

Another disadvantage of Kalman Filter-based solutions is their computational complexity which grows with $O(n^2)$, where n is the number of landmarks in the map $\mathbf{m}$. There are a number of methods that help reduce computational costs discussed in [20] but improvements can often lead to a loss of accuracy. An approach that aims to reduce computational complexity of feature-based EKF SLAM by reducing the number of maintained landmarks is presented in [46]. This reduction is claimed not to affect map consistency and efficiency if appropriate landmarks are chosen. A number of criteria for choosing which landmarks to maintain are presented but a poor choice of landmarks can lead to poor performance.

Divide and Conquer SLAM [125] is another method that aims to reduce computational costs per step from $O(n^2)$ to $O(n)$ but also improve the consistency of the predicted state. Instead of building a sequence of local maps of a large environment and then sequentially joining them together as is often done in the literature, it joins local maps in a binary tree fashion. This is a more compact representation that manages to reduce the computational complexity of mapping large environments while seemingly also yielding good consistency. The results presented have not been tested experimentally, however, and simplifications required to use this system in a real-world scenario may affect performance.

Kalman-based approaches do not account for the possibility of systematic bias errors that can lead to divergence. An EKF-based approach that uses neural networks to predict the uncertainty of the system due to a poor choice of model or extreme non-linearities is presented in [33]. This approach produces a filter that can theoretically adapt to changes in the environment or false model assumptions. No results of using this approach in a real system are presented, however, and if an unbiased vehicle model is used this method presents no advantages over using a simple EKF.

Finally, the performance of the filter depends greatly on the choice of the covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ for the pose and observation noise distributions. These values can be chosen empirically by assigning a larger observation covariance and smaller position covariance for a system that can predict position well but is equipped with sensors that are more prone to errors, for example. This sensitivity

to parameter choice means that an unsuitable choice of parameters can delay or prevent convergence.

A neurofuzzy-based adaptive Kalman filtering algorithm has been proposed which attempts to estimate the elements of the **R** matrix of the EKF algorithm at each sampling instant when a measurement update step is carried out. It thus aims to avoid consistency problems caused by inappropriate choices of the **Q** and **R** matrices. An improvement is observed over the traditional EKF for a simulated environment but complexity is added in order to estimate the **R** value [30].

An alternative linearisation to the EKF is the one produced by the Unscented Kalman Filter (UKF) which uses a weighted statistical linear regression process [81, 86]. Feature-based UKF SLAM has been used in large-scale outdoors environments [104] showing an improvement in consistency and accuracy in terms of prediction errors and uncertainty [81]. However, testing this method in real-world large-scale environments can be very computationally expensive if a large number of features is maintained and updated. A visual SLAM UKF implementation is presented in [77], where a Square Root UKF is used that helps avoid computationally costly mathematical manipulations [146]. This implementation manages to reduce the computational complexity of the UKF from $O(n^3)$ to $O(n^2)$.

Information filters

Information filters are subject to the same underlying principles as Kalman filters. The key difference between the two is how they represent Gaussian belief: while Kalman filters represent Gaussians by their moments (mean, covariance), Information filters represent them in their canonical parametrisation [134]. They use the information matrix defined as $\Omega = \Sigma^{-1}$ and the information vector defined as $\xi = \Sigma^{-1}\mu$, where Σ is the covariance and μ is the mean of the Gaussian.

The main advantages of this representation are improved numerical stability and the ability to represent global uncertainty easily (simply set $\Omega = 0$ rather than $\Sigma = \infty$). This filter can be computationally expensive since matrix inversions are needed to compute the information matrix and vector and it is therefore considered inferior to the Kalman filter for high dimensional spaces [134].

The Sparse Extended Information Filter [140] is a computationally efficient version of the Information filter that exploits the structure of the SLAM problem to represent maps through local, web-like networks of features allowing updates to be performed in constant time, irrespective of the number of features in the map. Given its computational savings this type of approach is well-suited to outdoors [97] and multi-robot exploration [137] and is used to update feature-based maps. This approach, however, produces error estimates that are overconfident when

expressed in the global reference frame according to [149] where an improved SEIF is proposed to address this problem.

Particle filters

Particle filters use samples to represent the posterior distribution instead of the parameterised representation used in Kalman and Information filters. They do not calculate the full posterior but use sets of samples to approximate it. As the number of particles approaches infinity this representation becomes equivalent to the actual posterior [135]. This means that the particle filter can approximate any distribution and does not require linear robot dynamics or Gaussian distributions. The sampling method used is Sequential Importance Sampling (SIS), a Monte Carlo method, as discussed in [15].

Using this method $\mathbf{x}_k$ is represented by a set of samples or particles sampled from

$$\mathbf{x}_k^{[n]} \sim p(\mathbf{x}_k^{[n]} | \mathbf{x}_{k-1}^{[n]}, \mathbf{u}_k) \tag{2.6}$$

for the nth particle, which can be used to perform the time update, Equation 2.3. In particle filtering $p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})$, used to perform the measurement update (Equation 2.2) is called the importance factor or weight

$$w_k^{[n]} = p(\mathbf{z}_k|\mathbf{x}_k^{[n]}, \mathbf{m}) \tag{2.7}$$

The particle filtering algorithm works as follows [135]. Particles are first initialised by sampling from the prior pose distribution $p(\mathbf{x}_0)$ (a more detailed discussion of how a prior is chosen will be presented in Section 2.3). Particles are then produced by sampling from the motion model, Equation 2.6. This set of particles constitutes the proposal distribution, $\bar{\chi}_k$. These are then weighted using weights proportional to $w_k^{[n]}$, Equation 2.7, to produce estimates of the posterior distribution χ_k as required, Equation 2.2. New particles are sampled from the set of weighted particles and these represent the approximation of the required posterior. A graphical explanation of the particle filter process can be seen in Figure 2.7 and a detailed mathematical derivation of the particle filter equations can be found in [15].

This method has a number of advantages. Unlike the Kalman filter-based solutions there is no need to assume linear relationships between inputs and states or Gaussan noise. It is also suited to both landmark and occupancy map representations without any need to process range sensor data to place them in a landmark format.

Particle filtering has a number of drawbacks, however, including degeneracy

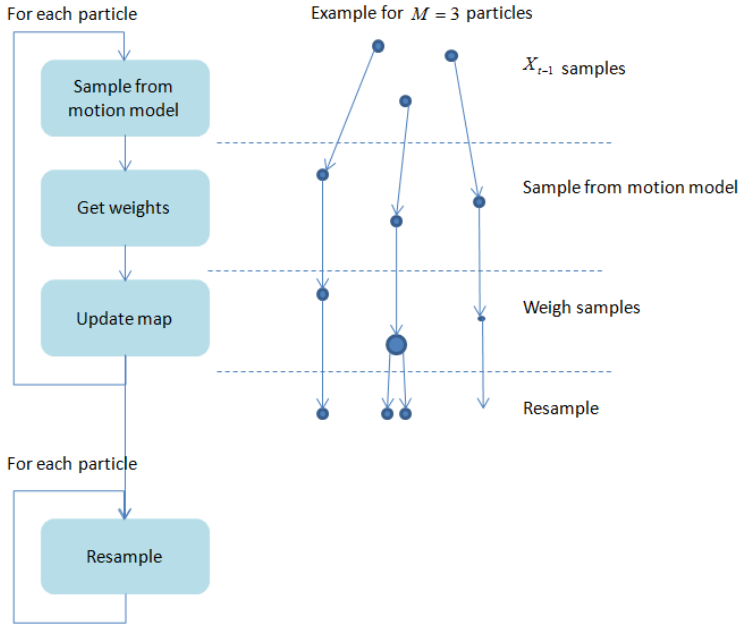


Figure 2.7: The particle filter process: Samples are taken from the motion model and relevant weights are determined. The map is then updated using that information and new particles are sampled from the set of weighted particles.

[15, 50] whereby all but one particle have negligible weights after a few iterations. One way of avoiding degeneracy is resampling: whenever degeneracy is observed particles associated with small weights are eliminated. This method, however, leads to particles with high weights being selected many times which leads to a loss of diversity in the final sample [18]. A discussion of more sampling methods can be found in [15]. Another drawback of using this method is the trade-off between increased accuracy and computational complexity determined by the number of particles used, since the more particles used the better the prediction accuracy but also the greater the computational cost.

Research such as [66] and [25] mathematically manipulates the proposal distribution to produce better samples. In [66] the proposal distribution is conditioned on the previous position and control input as well as the most recent observation to reduce estimation uncertainty. Selective re-sampling is also used to avoid particle depletion. The work presented in [25] takes samples from a proposal distribution conditioned on the whole trajectory rather than simply the state at the previous time step.

Others aim to improve the sampling method used. The research presented in [156] uses Tabu search to choose samples, aiming to avoid particle degeneracy and reduce the number of particles used. A different approach described in [27]

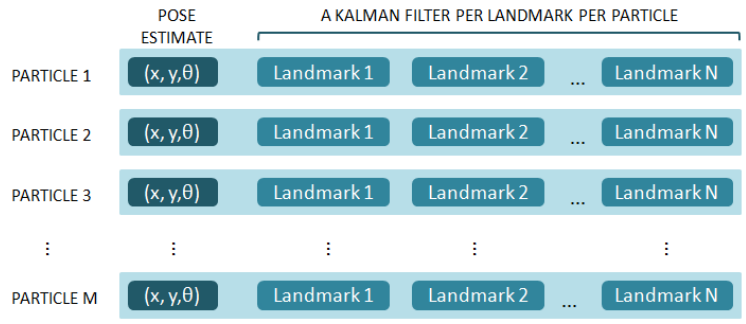


Figure 2.8: The FastSLAM solution: each particle holds estimates for all features in the map and each feature estimate is updated using a Kalman filter.

uses rejection sampling to reduce the estimation error. Finally, research such as [35] uses information from the innovation error to modify the number of particles used and thus improve performance.

FastSLAM A particle filter based method proposed to address the computational complexity drawbacks of particle filter SLAM is FastSLAM [110]. FastSLAM performs Rao-Blackwellization to decompose the SLAM problem into a robot localization problem and a collection of landmark/map estimation problems that are conditioned on the robot pose estimate

$$\begin{aligned} & p(\mathbf{X}_{0:k}, \mathbf{m} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \\ = \; & p(\mathbf{m} | \mathbf{X}_{0:k}, \mathbf{Z}_{0:k}) p(\mathbf{X}_{0:k} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \end{aligned} \tag{2.8}$$

In this case the aim is to compute the joint posterior of the map and the robot trajectory $\mathbf{X}_{0:k}$ rather than the single pose $\mathbf{x}_k$. That is because landmarks conditioned on the trajectory are independent, allowing the factorisation in Equation 2.8. The particle filter can then be used to sample from the motion model and produce the proposal distribution but now the estimates for the landmark locations (conditioned on the robot trajectory estimate) are performed separately.

Since the pose and map estimates can be performed separately using Rao-Blackwellization, a Monte Carlo Localisation (MCL) algorithm [139] can be implemented to produce pose estimates and calculate $p(\mathbf{X}_{0:k} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$. Then the mapping or landmark position estimate can be performed using an appropriate method. The particle filter can be implemented using both a feature-based and an occupancy grid map representation. Examples of the most commonly used versions of each are given below.

Landmark-based If a landmark-based map is updated then a Kalman filter is used for each of the N landmarks in the map to predict their position, Figure 2.8. This method would appear to be computationally expensive since $N \times M$ Kalman filters need to be maintained, where M is the number of particles used. However, a compact tree representation can be used to store updates that can lead to computational complexity that is logarithmic in the number of landmarks in the map-a vast improvement from the quadratic cost of Kalman filter-based solutions [110].

Occupancy-based In this case an occupancy map is updated for each particle [66, 69, 131]. Maintaining M maps (a map per particle) can also be computationally expensive. This issue is addressed by the commonly used DP SLAM which uses a tree structure to store maps efficiently and reduce computational complexity to a worst case of log-quadratic [53]. DP SLAM 2.0 relaxes the assumptions of a perfect laser range sensor made by DP SLAM to produce a solution that works better using real sensors [52].

Finally, a number of variations such as Unscented FastSLAM [87] look at improving the sampling method for the proposal distribution to avoid the problems of degeneracy and improve performance.

The filtering algorithm used throughout this thesis is occupancy grid FastSLAM. Occupancy grid SLAM uses an occupancy grid map representation which allows prior information to be easily incorporated. DP SLAM does not maintain a grid map updated using the occupancy grid mapping algorithm [51] but instead updates and maintains a tree of occupancy values for all particles based on sensor readings rather than an iterative update process for each square. Gmapping [66] can be used in conjuction with DP SLAM to leverage the improved motion model of gmapping that incorporates sensor information to improve motion model predictions and the computational efficiency of DP SLAM. These two do not have to be combined, however and the map can be updated using the occupancy grid mapping algorithm as discussed in [51] to allow the inclusion of prior information.

Smoothing or pose graph approaches

The filtering methods described in this section all process then discard sensor and odometry data, which means data cannot be revisited when constructing a map [138]. A category of methods that aim to overcome this issue are smoothing or pose graph techniques. These are performed offline, storing all collected data and only performing mapping at the end of exploration [99].

The posterior of this formulation, the full SLAM problem, is represented as a sparse graph. This graph is subject to a number of nonlinear quadratic constraints and optimising these constraints yields a maximum likelihood map and a corresponding set of robot poses [99].

A commonly implemented offline technique is Graph SLAM [99], which uses sparse constraint graphs representing soft constraints extracted from the data. The map and robot path are obtained by using these constraints to produce a globally consistent estimate. The constraints are commonly nonlinear so they are linearised resulting in a least squares problem which can be solved using common optimisation techniques.

A different method to optimise graph-based nonlinear error functions is presented in [94] as g^2o, a general framework for graph optimisation that can be used for offline SLAM methods. This framework can be applied to any non-linear least squares problem represented as a graph and presents an approach that reduces the computational requirements of previous approaches. It yields improved results by exploiting the graph sparsity and using a new error function, allowing these techniques to be used for 2D as well as 3D graph-based SLAM.

Another approach that uses factor graphs is Georgia Tech Smoothing And Mapping (GTSAM) [45]. Factor graphs are a different representation of Bayes networks for a Hidden Markov Model (HMM), leveraging the fact that measurements are known. Thus the factor graph only represents the unknown variables connected to factors that contain probabilistic information about them, derived from the Markov chain or the observation model [45]. A Maximum A-Posteriori (MAP) inference is performed to maximise the value of the factor graph. The factor graph specifies the posterior over the entire trajectory of the robot, rather than just the last pose, making this is a smoothing approach. This approach can be used in pose SLAM, landmark-based SLAM and structure from motion but is not well suited to occupancy grid SLAM.

These approaches are not well suited to occupancy grid map representations; as such, they are not the implementation of choice for this thesis.

Commonly used approaches and remaining challenges

The Bayesian formulation of SLAM was introduced in the 1990s. The SLAM filtering algorithms that are commonly used today and are considered established, state-of-the-art solutions presented in Section 2.2.3 were proposed in the early 2000s, with FastSLAM [110] first introduced in 2002 and DP SLAM [52] introduced in 2004. Since then most research into robot localisation and mapping has focused on minor improvements in terms of more efficient sampling, exploring

newer sensors such as the Microsoft Kinect [72], exploring 3D mapping [130] and improving performance for robots exploring dynamic or cluttered environments [78].

SLAM is commonly considered a theoretically solved problem but SLAM solutions do not work reliably in changing, challenging real-world environments. The focus of localisation and mapping applications has recently shifted to self-driving cars [68, 120]. Some of the methods used for self-driving car navigation build on the field's established techniques [37, 115, 129] but no detailed technical papers on how such cars operate have been published due to the commercial nature of such applications.

Commonly used SLAM implementations can be found on the OpenSLAM website and as ROS (Robot Operating System) [3] modules. The occupancy grid-based SLAM solutions in these sources are DP-SLAM [52], gmapping [66], GridSLAM [69] and Hector SLAM [90] for 2D occupancy grid maps and OctoSLAM [57] for 3D maps. SLAM code available on OpenSLAM can be outdated and much of it is not maintained.

Gmapping is the most commonly used occupancy grid SLAM implementation in ROS. This implementation assumes grid squares can either be occupied, unoccupied or unknown and uses sensor data to update occupancy values. It thus only needs to maintain information about explored areas as they get explored. These assumptions may apply to expensive sensors or in environments where sensors can perform reliably. They cannot be trusted as much, however, if inexpensive, inaccurate sensors are used or if the robot operates in an environment where sensors can perform unreliably such as USAR post-disaster areas. It is therefore important not to rely on the latest sensor reading to update occupancy values. Similarly, DP SLAM assumes perfect sensors, relaxing this assumption for DP SLAM 2.0.

In order for robots to be valuable in challenging real life applications they need to be able to perform reliably even if low-cost sensors are used or sensor data becomes unreliable. That is why this thesis proposes the use of optimised Bayesian priors to incorporate available prior information to the SLAM algorithm. Using this information to construct a Bayesian prior for recursive mapping reduces the reliance on sensor data and allows improved map quality without the need to alter the SLAM algorithm used. The proposed priors can be used for any SLAM implementation that uses a probabilistic recursive algorithm to update occupancy maps.

2.3 SLAM priors

In the solutions presented in the literature to date SLAM improvements have been attempted by proposing new filtering methods, sensors or map representations. Little research has been conducted, however, in terms of studying how meaningful priors can improve performance. This thesis addresses this problem by proposing a method to construct informative priors that improve the performance of SLAM without increasing computational complexity. Unlike other research that uses prior information in feature-based SLAM, often to constrain landmark locations, this thesis proposes a method to construct an informative Bayesian prior for occupancy grid based implementations. It explores this concept by presenting a case study for indoors occupancy grid SLAM.

2.3.1 Prior information in robotics

The use of prior information to improve the performance of robotics systems has been suggested for a number of systems that operate in real life challenging environments. Priors are used to improve navigation, localisation and mapping of self-driving cars such as using experience to improve localisation [101] and using 3D semantic priors to interpret traffic lights [24]. They are also often used to improve the performance of odometry predictions. In visual odometry implementations [12] a relative pose prior can be constructed by tracking feature movements between frames to improve pose prediction accuracy. The use of prior knowledge of motion preferences has also been proposed to improve the quality of Bayesian occupancy filtering in dynamic environments [63].

Prior knowledge is incorporated in some implementations of feature-based SLAM where the landmarks in the environment are known [28]. This knowledge simplifies the problem and converts it into matching detected features to the list of a priori known landmarks. In order to avoid having to maintain a large lookup list, implementations such as [153] use only a few known landmarks as points of reference to correct predictions. Skeletal SLAM [108] uses prior information about the shape of a building to construct more accurate occupancy grid maps.

Other implementations constrain the possible location of landmarks in GraphSLAM [124] using prior information. As a result, the observed position error is reduced whilst maintaining the same or improved consistency compared to the no-prior solution [42]. A method to incorporate prior information into outdoors EKF-based SLAM is presented in [123] yielding improved performance. An implementation that uses aerial images to extract prior information and improve the performance of outdoors SLAM by inserting correspondences between stereo

and 3D range sensor data and the aerial images as constraints into a graph-based SLAM implementation is presented in [89] and is found to improve map consistency.

In all of the above cases using prior information was found to be beneficial and yield improved performance. Prior information has mostly been used to facilitate feature, object or pose identification. However, none of these implementations have attempted to use this prior information to construct a Bayesian prior map. Using all available information to produce more realistic and more accurate maps without adding to the computational cost of the SLAM algorithm can make a significant contribution towards useful SLAM solutions for USAR, achieving the requirements detailed in Section 2.1.3. The concept of using available prior information to construct informative Bayesian priors for SLAM is an area that has not been researched and is addressed in this thesis.

2.3.2 Bayesian priors in SLAM

SLAM performs a Bayesian estimation to determine the robot's pose and a map of the environment given sensor readings, control inputs and the robot's initial pose. Formally, it aims to compute the joint posterior of the robot's pose $\mathbf{x}_k$ and the map $\mathbf{m}$ [50]

$$p(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \propto \mathbf{L} \times p(\mathbf{m}) \tag{2.9}$$

for all times k, where $\mathbf{x}_k$ is the state vector describing the robot's pose, $\mathbf{m}$ is the map of the environment (or a selection of landmarks for feature-based SLAM), $p(\mathbf{m})$ is the prior map, $\mathbf{Z}_{0:k}$ are the sensor observations, $\mathbf{U}_{0:k}$ the history of control inputs, $\mathbf{x}_0$ is the robot pose at time $k = 0$ and $\mathbf{L}$ is given by

$$\mathbf{L} = \frac{p(\mathbf{z}_k, \mathbf{x}_k | \mathbf{m})}{p(\mathbf{x}_k, \mathbf{m})} \frac{p(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{p(\mathbf{z}_k | \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \tag{2.10}$$

Since the joint posterior in (2.9) can be written

$$\begin{aligned} & p(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \\ & p(\mathbf{z}_k | \mathbf{x}_k, \mathbf{m}) \frac{p(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{p(\mathbf{z}_k | \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \\ = & \frac{p(\mathbf{z}_k, \mathbf{x}_k | \mathbf{m})}{p(\mathbf{x}_k, \mathbf{m})} p(\mathbf{m}) \frac{p(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{p(\mathbf{z}_k | \mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \\ & \propto \mathbf{L} \times p(\mathbf{m}) \end{aligned} \tag{2.11}$$

indicating that the joint posterior is proportional to the prior probability $p(\mathbf{m})$ and

as such a choice of an optimal prior $p(\mathbf{m})$ can lead to a more accurate estimate of the joint posterior.

In the majority of indoors SLAM solutions the environment to be explored is assumed to be unknown [50, 53, 109] and an uninformative prior $p(\mathbf{m})$ is assigned. Some researchers aim to produce solutions that do not depend on having prior knowledge of the environment [20]. Others claim that access to detailed architectural drawings or other sources of prior information for outdoors implementations may be difficult [93]. The hypothesis examined in this thesis is that constructing and using appropriate informative priors can improve the performance of existing SLAM algorithms. In order to test this hypothesis indoors occupancy grid FastSLAM is the case study investigated in this thesis.

2.3.3 Occupancy grid FastSLAM priors

In occupancy grid FastSLAM [109] the process of updating pose estimates and updating the map can be performed separately as shown in Equation 2.8. Mapping is performed using the occupancy grid map algorithm [51] and the predicted poses. This SLAM format allows for an analysis of the mapping algorithm where prior information is incorporated through the use of a matrix of prior probabilities for each grid cell, $p(\mathbf{m})$.

The occupancy grid mapping algorithm [51] splits the environment into a grid of cells and a prior probability is assigned to each cell. Cells are assumed to be independent in order to update the probability of occupancy of each grid cell. This assumption allows the factorisation

$$p(\mathbf{m}|\mathbf{X}_{0:k}, \mathbf{Z}_{0:k}) = \prod_i p(\mathbf{m}_i|\mathbf{X}_{0:k}, \mathbf{Z}_{0:k}) \tag{2.12}$$

where $i = 1, ..., N$ and N is the total number of grid cells, making this an easier estimation since updating the probability for each cell given the robot pose and sensor readings is merely an update of the probability of occupancy of that cell. As the robot moves through the environment the probabilities of occupancy of cells that are within the field of vision of the robot sensors are updated.

The probability that a cell $\mathbf{m}_i$ is occupied given the observation history is given

by

$$\begin{aligned} p(\mathbf{m}_i|\mathbf{Z}_{0:k}) &= \frac{p(\mathbf{z}_k|\mathbf{m}_i, \mathbf{Z}_{0:k-1})p(\mathbf{m}_i|\mathbf{Z}_{0:k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{0:k-1})} \\ &= p(\mathbf{z}_k|\mathbf{m}_i)\frac{p(\mathbf{m}_i|\mathbf{Z}_{0:k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{0:k-1})} \\ &= \frac{p(\mathbf{m}_i|\mathbf{z}_k)p(\mathbf{z}_k)}{p(\mathbf{m}_i)}\frac{p(\mathbf{m}_i|\mathbf{Z}_{0:k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{0:k-1})} \end{aligned} \tag{2.13}$$

Placing Equation 2.13 in the odds form we get

$$\begin{aligned} odds(\mathbf{m}_i|\mathbf{Z}_{0:k}) &= \frac{p(\mathbf{m}_i|\mathbf{Z}_{0:k})}{p(\neg\mathbf{m}_i|\mathbf{Z}_{0:k})} \\ &= \frac{p(\mathbf{m}_i|\mathbf{z}_k)p(\mathbf{m}_i|\mathbf{Z}_{1:k-1})p(\neg\mathbf{m}_i)}{p(\neg\mathbf{m}_i|\mathbf{z}_k)p(\neg\mathbf{m}_i|\mathbf{Z}_{1:k-1})p(\mathbf{m}_i)} \\ &= odds(\mathbf{m}_i|\mathbf{z}_k)odds(\mathbf{m}_i|\mathbf{Z}_{1:k-1})(odds(\mathbf{m}_i))^{-1} \end{aligned} \tag{2.14}$$

where $p(\mathbf{m}_i)$ is the prior probability of occupancy of the ith grid cell, $odds(\mathbf{m}_i|\mathbf{z}_k)$ represents the inverse sensor model and $odds(\mathbf{m}_i|\mathbf{Z}_{1:k-1})$ is the odds at the previous time step. Taking the logarithm of Equation 2.14, the log odds representation of Equation 2.14 is

$$l_{k,i} = InverseSensorModel(\mathbf{m}_i, \mathbf{x}_k, \mathbf{z}_k) + l_{k-1,i} - l_{0,i} \tag{2.15}$$

with

$$l_{k,i} = log\frac{p(\mathbf{m}_i|\mathbf{Z}_{1:k}, \mathbf{x}_{1:k})}{1 - p(\mathbf{m}_i|\mathbf{Z}_{1:k}, \mathbf{x}_{1:k})} \tag{2.16}$$

Research has been conducted to identify suitable or preferable sensor models such as [136] and [157].

The Bayesian priors for each cell, $p(\mathbf{m}_i)$ are incorporated through $l_{0,i}$, the log odds prior for the ith cell. When there is no available prior information a non-informative Bayesian prior is assigned to all grid cells, $p(\mathbf{m}_i) = 0.5,\ i = 1, ..., N$.

In real life time-critical applications using an appropriately constructed prior can lead to a more accurate map even after a few time steps. Incorporating prior information does not add to the computational cost of running SLAM and only incurs a one-off cost of extracting prior information and constructing a prior. Since the posterior is proportional to the prior map, Equation 2.9, constructing an optimised prior map $p(\mathbf{m})$ can improve performance, especially for time critical applications where the environment needs to be explored quickly, making the effect of the prior more significant. Therefore this thesis proposes the use of architectural drawings to extract structural information and construct optimised indoors

Bayesian priors.

2.3.4 Ideal prior properties

The way in which occupancy grid prior information is incorporated into occupancy grid SLAM was discussed in the previous section. This section aims to explore desirable prior values.

The log odds representation is used to update the occupancy of each grid cell, plotted in Figure 2.9. A negative log odds value corresponds to a probability of occupancy below 0.5 and a positive log odds to a probability of occupancy above 0.5:

$$l(p) = \begin{cases} = 0, p = 0.5 \\ > 0, p > 0.5 \\ < 0, p < 0.5 \end{cases} \tag{2.17}$$

where p is the probability of occupancy and $l(p)$ the corresponding log odds, showing the relationship between the log odds value and the probability of occupancy it corresponds to. There are three elements that affect the updated value of occupancy: the previous occupancy, the prior and the inverse sensor model.

The log odds representation is used with the initialisation of $\mathbf{l}_{k,i} = \mathbf{l}_{0,i}$. Using Equation 2.15, this leads to the update rule

$$\begin{aligned} \mathbf{l}_{1,i} &= InverseSensorModel(\mathbf{m}_i, \mathbf{x}_1, \mathbf{z}_1) \\ \mathbf{l}_{2,i} &= 2 \times InverseSensorModel(\mathbf{m}_i, \mathbf{x}_1, \mathbf{z}_1) - \mathbf{l}_{0,i} \\ &\vdots \\ \mathbf{l}_{k,i} &= k \times InverseSensorModel(\mathbf{m}_i, \mathbf{x}_1, \mathbf{z}_1) - (k-1) \times \mathbf{l}_{0,i} \end{aligned} \tag{2.18}$$

for the ith cell at the kth iteration, when observing the same cell i times. Therefore the value of the updated occupancy depends on the relationship between the inverse sensor model and the prior, $\mathbf{l}_{0,i}$. Using Equation 2.19, to obtain a larger final map probability for empty cells we need $\mathbf{l}_{k,i} > 0$ corresponding to a probability of occupancy $\mathbf{p}^k_{m_i} > 0.5$, Figure 2.9. Therefore we would ideally want

$$\begin{cases} \mathbf{l}^{space}_{k,i} > 0 \\ \mathbf{l}^{wall}_{k,i} < 0 \end{cases} \tag{2.19}$$

$\mathbf{l}_{k,i} > 0$ would require

$$InverseSensorModel(\mathbf{m}_i, \mathbf{x}_k, \mathbf{z}_k) > \mathbf{l}_{0,i} \tag{2.20}$$

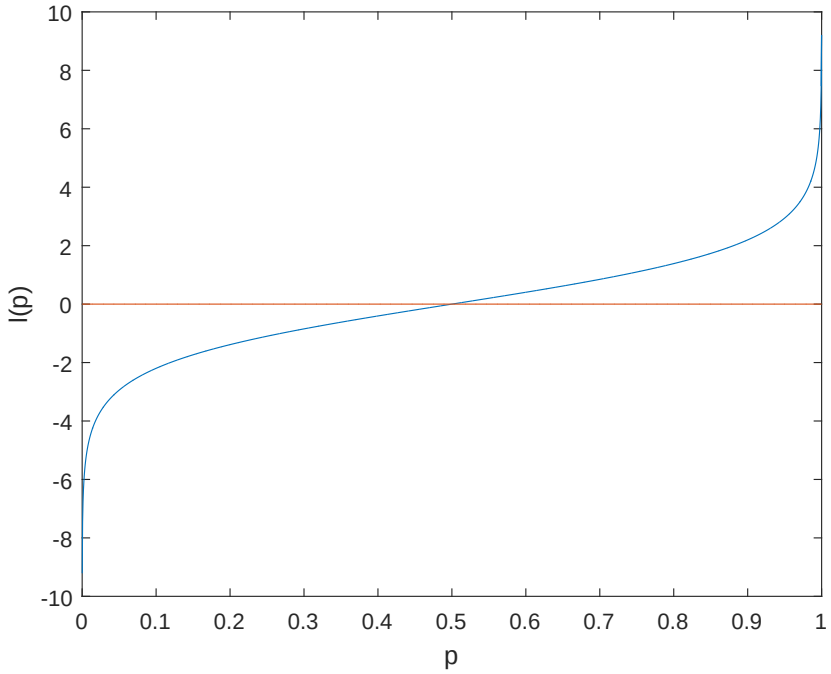


Figure 2.9: The log odds plot for probability values 0-1. In the region 0.2-0.8 the log odds show linear behaviour, whereas for values outside that range the resulting log odds value increases rapidly.

and $\mathbf{l}_{k,i} < 0$ requires

$$InverseSensorModel(\mathbf{m}_i, \mathbf{x}_k, \mathbf{z}_k) < \mathbf{l}_{0,i} \tag{2.21}$$

for a large k. Therefore the probability assigned by the sensor model to detected free space should be greater than the prior value assigned to empty space and the value assigned to detected occupied space by the sensor model should be smaller than the prior.

$$\begin{cases} p^{sensor}_{free} > p_{space} \\ p^{sensor}_{occupied} < p_{wall} \end{cases} \tag{2.22}$$

Depending on the sensor model used appropriate prior values should be bounded accordingly.

The log odds curve has two asymptotes near p values of 1 and 0 as expected, where it reaches infinity. Regions near the asymptotes should be avoided to ensure incorrectly assigned priors can be corrected and values smaller than 0.1 or greater than 0.9 should not be used where possible. The behaviour of the log odds is

nearly linear for $0.2 \leq p_{prior} \leq 0.8$. Therefore, where the inverse sensor model values allow it, prior values within that range should be chosen. Prior probabilities near 0.5 correspond to $l(p) = 0$, in which case the inverse sensor model dominates the update.

The effects of different prior values are examined in a quantitative manner in Section 4.4.

2.3.5 Prior information requirements for USAR

There are a number of possible approaches to constructing the prior map $p(\mathbf{m})$, with a non-informative prior which assigns 0.5 to all cells being the most common. Another approach that does not require detailed knowledge of the environment is to assign a prior value between 0 and 1 to all grid cells.

Using a value $p(\mathbf{m}_i) \in (0, 1)$ for all grid cells requires knowledge of how cluttered an environment is expected to be. Lower prior values such as 0.3 can be assigned to all grid squares for environments that are expected to be very cluttered and higher ones such as 0.8 for environments that are expected to be empty. Most environments contain significantly more empty space than walls and so it is preferable to assign a higher value if no detailed information is available. Assigning the same prior probability to all cells using information about the environment to decide which value in the range $(0, 1)$ is most suitable fails to provide information about unexplored areas.

A more rigorous approach would be to use available sources of raw prior information such as architectural drawings and floor plans to extract the location of elements such as walls. A first attempt would be to use a drawing as-is and assign a low prior value to all black pixels and a high prior value to all light pixels. For non-black-and-white drawings this approach can yield poor results since a binarisation would be required. Deciding on a threshold is non-trivial since walls can be represented using both grey and black lines. One way to overcome this would be to assign high prior values to all white pixels in the image and low ones to all non-white pixels. This approach, however, could lead to a great number of artifacts, especially for low quality images.

The main requirement when constructing a prior map for a robot deployed in a USAR mission is to ensure all information about wall locations is as accurate as possible. If a wall is missed in the prior map the information provided will be no worse than that provided by a non-informative prior. If, however, walls are expected where there are none confusion can ensue, with human rescuers being unable to easily and correctly interpret the map. This is especially important if parts of the map are unexplored and the prior map is used to provide information

about those areas. Therefore including correctly identified walls in the prior is more important than including all walls.

Given the above requirements, a method that extracts walls ensuring a very low number of false positives and a high precision would be required for a USAR-type scenario. A novel method to process such drawings and extract structural information is presented in the next chapter, where relevant metrics are used to ensure the high precision and low false positive requirement.

2.4 Summary

This chapter has presented the USAR environment and the value robots can add to rescue missions, identifying remaining challenges. The focus of this thesis is Simultaneous Localisation And Mapping (SLAM), one of the key remaining challenges. Whilst many solutions for SLAM have been proposed throughout the years the majority of solutions perform poorly in challenging real life scenarios such as USAR operations. Attempts have been made to improve performance through use of better sensors, filtering algorithms and map representations. Hardly any research has been conducted, however, to explore the benefits of different SLAM priors.

A main shortcoming of existing methods is that they often overestimate sensor accuracy in difficult real life scenarios or available computational resources and hence produce solutions that struggle in applications such as USAR. Using available prior information to improve the performance and reliability of robotics systems is common in the literature but no study of how constructing an appropriate contextual Bayesian prior has been conducted to date. This is a gap in the literature that is addressed by this thesis, aiming to produce a computationally efficient SLAM solution that produces accurate maps. In order to extract priors for indoors SLAM this thesis proposes the use of architectural drawings or floor plans and a novel method to process such drawings is presented in the next chapter.

Chapter 3

Using architectural drawings to extract prior information

The potential benefits of constructing an informative Bayesian prior using information about the environment that is available a priori were discussed in the previous chapter, highlighting the potential for improved performance without adding to the computational cost of the SLAM algorithm. There is, however, a one-off cost associated with extracting prior information and converting it to a prior format. This chapter addresses the problem of processing images containing prior information to extract structurally relevant elements that can then be used to construct Bayesian priors.

In order to construct informative contextual priors and test potential performance improvements a source of prior information needs to be identified. This can be an image such as an architectural drawing or floor plan for indoors SLAM or an aerial photograph or road network map for outdoors SLAM. Architectural drawings or floor plans are produced for all buildings, with standardisation regulations such as Building Information Management (BIM) [7] requiring digitised architectural drawings to be produced and be readily available for all buildings. Moreover, such drawings contain structural information such as the location of doors, walls and windows in the building and are drawn to scale, making them a very suitable choice for a source of prior information for indoors SLAM.

For indoors environments a floor plan or architectural drawing can be used as a source of prior information, with occupied sections corresponding to walls and unoccupied sections to empty space. Similarly, for an outdoors environment the source of prior information can be a road map, with empty space corresponding to roads and occupied space corresponding to non-traversable areas.

This thesis focuses on the use of architectural drawings as a source of informa-

tion and so this chapter presents a method to extract structural information from them that can be used to construct a prior map. Using a drawing without processing can lead to poor localisation quality and provide misleading or incomplete information about unexplored or partially mapped areas.

This chapter presents a novel strategy to process architectural drawings and floor plans using a simple set of geometric tests to extract structural information. Unlike the processing strategies found in the literature it prioritises wall detection; detected walls can then be used to construct SLAM priors. A set of criteria by which the quality of constructed prior maps can be assessed is also proposed. Once this information has been extracted it can be placed in an appropriate prior map format as will be discussed in Chapter 4. The research presented in this chapter was published in the proceedings of the 6th International Conference on Automation, Robotics and Applications (ICARA) 2015 [59].

The contributions of this chapter are as follows:

- A novel method for door and wall detection in architectural drawings and floor plans
- A novel method to automatically produce indoors maps that can be used for robot localisation
- A method to assess the quality of structural information extracted from architectural drawings by formulating the image processing problem as a binary classification of pixels as walls or empty space

This chapter is organised as follows. Section 3.1 defines the types of drawings examined as well as the benefits of using such drawings as a source of prior information and presents an overview of methods used in the literature to process such drawings. These methods are found to be ill-suited to extracting structural information to construct informative priors, identifying a gap in the drawing processing literature. Section 3.2 details the requirements for a drawing processing algorithm used to extract prior structural information. Section 3.3 details the proposed novel algorithm that uses geometric constraints to detect doors in the image and uses their relation to walls to detect building walls. A novel approach to performance assessment is also presented, viewing the problem of wall detection as a binary classification of image pixels as belonging to either walls or empty space. Finally, Section 3.4 presents the results of using the proposed drawing processing algorithm, with the proposed method yielding a precision of at least 98%, recall of at least 61% and a false positive lower than 0.09 % for a set of representative drawings. Section 3.4 also benchmarks the proposed approach against a com-

monly used wall detection method, dilation, and the proposed approach is found to yield good results across all drawings unlike dilation.

3.1 Architectural drawing background

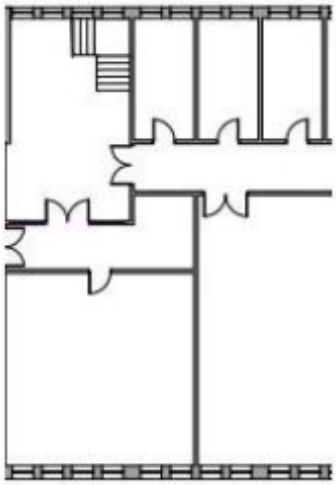
The types of drawings considered in this thesis are presented in this section and their suitability as a source of prior information is discussed. An overview of drawing processing methods commonly used in the literature is also given, highlighting the shortcomings of currently used methods and the need for a drawing processing algorithm that is more suited to prior construction.

3.1.1 Architectural drawings as a source of prior information

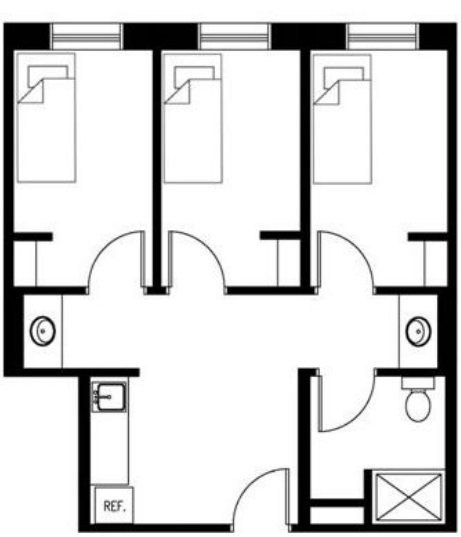
Architectural drawings or floor plans within this thesis are defined as 2D, hand or computer drawn top-down drawings of buildings, containing structural information, suggested furniture and annotating text. Architectural drawings are drawn to scale and contain more detail than floor plans, annotations and more strictly defined symbols and representations, Figure 3.1(a). Floor plans on the other hand often contain more suggested furniture and room labelling and do not always contain details such as different representations for different wall types or materials used, for example, Figure 3.1(b).

Architectural drawings and floor plans are an ideal choice for a source of indoors prior information since they contain structural information about the building such as the location of walls, doors and windows and are drawn to scale, providing information about the size of the environment. This structural information can be used to construct a prior map of the environment. Moreover, such drawings are produced for all buildings and are available for most buildings. With the introduction of BIM in 2012, storing drawings of all buildings [7] in the UK is becoming a requirement, making access to digitised drawings easier and more commonplace. Finally, there are no other sources of prior information for indoors environments that can provide structural information drawn to scale that are also readily available, making such drawings a suitable source of prior information.

Given the large amount of information contained in such drawings there is a need to process them to differentiate between types of information such as wall locations, door locations and annotating text, for example, and extract information that can be used to construct an appropriate prior map. Indoors SLAM maps commonly depict structural information, most commonly wall locations, and thus architectural drawings need to be processed to extract wall locations.



(a)



(b)

Figure 3.1: Examples of an architectural drawing (a) and a floor plan (b). The architectural drawing contains more detail when it comes to walls and stairs whereas the floor plan contains more suggested furniture.

Architectural drawings are commonly produced using specialist software but they are reproduced, distributed and accessed as images which need to be processed to differentiate between different types of symbols and features. Image processing, computer vision and machine learning techniques have therefore been employed to extract information from architectural drawings and floor plans, as discussed in the next subsection. These algorithms were developed for 2D to 3D drawing conversion [100, 119], better interpretation of hand-drawn symbols [31] or drawing archiving [10].

3.1.2 The need to process drawings and extract information

When using an architectural drawing or floor plan as a source of raw prior information a method needs to be devised to convert the information in the drawing to a prior format. In an occupancy grid map representation we can use the architectural drawing to assign an informative value to each grid cell instead of assigning 0.5 to all grid cells. We can assign a lower prior probability to areas that correspond to occupied space in the drawing and a high probability to areas that correspond to empty space.

Given the prior is only a starting point for recursive mapping a natural first step is to examine whether a drawing needs to be processed at all, or whether the raw drawing can be used as-is to assign low prior probabilities to dark pixels in the drawing and high prior probabilities to white areas in the drawing. In order to do so greyscale or coloured drawings need to be binarised, so some processing is required for all non black-and-white drawings.

The binarised map is used to localise the robot. Using the drawing as-is can lead to localisation errors, especially when the drawing contains large annotating text or lines. It is therefore important to provide a map that is as accurate a representation of the building's structure as possible. The proposed method to extract walls and empty space can also be used to automatically construct indoors maps for robot localisation and experiments using a drawing as-is and using extracted walls and empty space are presented in Chapter 6.

The prior map also provides information about partially mapped or unexplored areas and this information needs to be as easy to understand and interpret as possible. Annotating lines in the prior drawing that appear in a partially mapped room may be wrongly assumed to correspond to walls or in-built furniture and mislead rescuers, for example.

Extracting walls and empty space from drawings could be performed by hand by a human operator. Using an automated method like the one we propose has a number of advantages, however. Performing the extraction automatically:

- Frees up time for first responders to perform other tasks
- Avoids mistakes by human operators under pressure
- Does not require users to know the format and conventions used in architectural drawings

The following section explores existing automatic drawing processing solutions.

3.1.3 Architectural drawing processing background

Floor plans for buildings such as hospitals and offices are generally available and can be used to construct indoors SLAM priors. Whilst they contain useful information such as the location of the building's walls they also contain text and suggested furniture. They thus need to be processed to extract structural information in order to construct useful priors.

Processing drawings to extract information has been of interest in recent years due to the introduction of BIM [7] standards which require 3D models as well as 2D drawings to be available for buildings. 2D drawings of older buildings will thus need to be processed to produce 3D models. A number of image processing, computer vision and machine learning techniques have also been used to extract information from architectural drawings, for use in applications such as electronic archiving of data.

Some research such as [9] uses image processing and computer vision techniques like Canny Edge detection or the Hough Transform to detect symbols in architectural drawings. This type of approach does not generalise well, however, since there are no universal symbols for elements such as doors, Figure 3.2, and no standardisation of drawing rules [17]. Rule-based approaches as well as machine learning techniques have also been used to detect hand drawn symbols [16, 85, 92]. In [118] neural networks are trained to recognise commonly occurring feature shapes. These require a representative set of drawings to train the machine learning algorithm but such a set is hard to obtain given the wide variety of possible symbol representations [17].

A different approach followed by [10, 11, 100] proposes the use of image processing techniques like dilation and erosion to distinguish between thick and thin lines in the image. Different line thicknesses are then associated with different drawing features. This technique often performs poorly since drawings use varying line thicknesses for elements in the image, especially in the case of floor plans where wall and door thickness does not need to be drawn to scale. The drawings

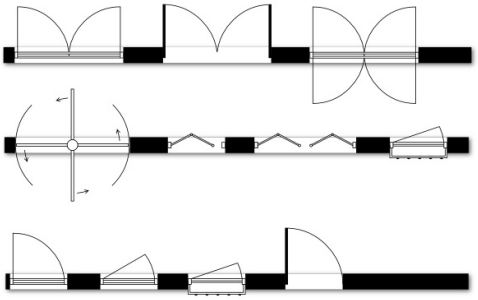


Figure 3.2: Different door representations used in floorplan creating software from the floorplanner.com website showing a examples of possible door representations.

used in [10] and [100] have been produced by the same architectural office which causes further concerns about how well their method can generalise.

An alternative representation of drawing information is proposed in [100, 119], where the information in the image is vectorised and polyline information obtained after vectorisation is used to detect walls. The aim of this research is 3D reconstruction using a 2D drawing and as such the results presented are 2D to 3D conversions, making it difficult to evaluate the merit of this approach in terms of reliable 2D feature extraction.

Some attempts have also been made to create networks of constraints to interpret architectural drawings after vectorising them. One of them aims to recognise all possible symbols by allowing users to define and update a dictionary of possible symbols [8]. This type of method depends greatly on the quality of the vectorisation and needs to be updated constantly to include new symbols.

Interpretation of hand-drawn symbols in real time, as they are being drawn, is proposed in [31], whereby a maximum likelihood analysis is used to determine which symbol is most likely being drawn. Existing databases of symbols are used to attempt to match the symbol drawn to a symbol in the database. This approach depends greatly on the examples included in the database and assumes that symbols are being drawn one after the other, focusing on processing individual symbols rather than the overall drawing.

In all of the above methods the main aim is to extract information and present it in a human-friendly format. This has led to a natural emphasis on extracting drawing features that are not necessarily structurally relevant, such as room-labelling text. Conversely, these algorithms do not prioritise features such as walls, which are essential for robot navigation. These methods are therefore not well suited to constructing SLAM priors, motivating the need for a drawing processing method that prioritises accurate wall detection across different types of drawings and sym-

bol representations.

3.2 Problem formulation

The main applications of processing architectural drawings and floor plans to extract information are 2D to 3D drawing conversion and archiving. In all these cases the problem has been viewed as a pattern processing one, aiming to identify selected symbols as accurately as possible. The information extracted has been converted into a human-friendly format and metrics used to assess performance take this focus into account. If the extracted information is to be used to construct a SLAM prior, however, it needs to be converted to a robot-friendly format. When extracting information to construct a SLAM prior and a map to be used for robot localisation the main priority is identifying wall locations. Walls give the outline of the building and can be easily incorporated into a robot-friendly format such as an occupancy grid map.

The main challenges faced when trying to process an architectural drawing or floor plan are [17, 32]:

- Great variability between drawings produced by different architects or using different software [158]
- Large number of possible representations for elements such as doors (Figure 3.2)
- Intractable number of acceptable designs
- Lack of strictly defined drawing rules

Unlike other types of drawings the design rules in architectural drawings are numerous and can often be bypassed as long as the building drawn meets health and safety regulations. For example, door sizes are not restricted as long as the doors are easy to move through and furniture can fit through them [32].

The problem addressed in this chapter is how to process any architectural drawing or floor plan to extract useful structural information for robot navigation that can be converted into a SLAM prior format.

The key requirements are:

1. A method that performs well for any floor plan or drawing: there are no universal symbols for elements such as doors and a number of slightly different representations can be used [17, 32]

2. Detecting walls: simply assigning dark pixels as walls causes false positives such as text to be used as wall points
3. Converting detected walls to a SLAM prior: the chosen map representation needs to translate well from drawing to SLAM prior
4. Determining an appropriate set of criteria to assess the quality of the extracted prior information

Given this problem description the chosen approach should make use of existing information about symbols and drawing rules without tailoring the solution to a certain architect's drawing preferences and style. However, it should also avoid a method that would require building a constantly growing and eventually intractable dictionary of potential symbols. The approach proposed in this thesis uses simple rules to detect families of features leveraging the geometric characteristics of commonly used symbols to produce a drawing independent algorithm.

3.3 Proposed approach

This section presents a novel approach to process architectural drawings and floor plans and extract structural information that can be used to construct prior maps. Unlike the approaches proposed in the literature it prioritises the accurate detection of walls whilst also ensuring the method used performs well for different drawing styles. A novel approach to assess drawing recognition quality is presented, viewing the problem of accurate wall detection as a binary classification of all pixels in the image as wall or empty space points.

3.3.1 Drawing processing

In order to decide on an appropriate processing algorithm a target robot map representation needs to be chosen and occupancy grid maps [51] are chosen for this thesis. Occupancy grids are chosen because the incorporation and visualisation of prior information is relatively straightforward. The map at time $k = 0$ is the Bayesian prior constructed based on available information and if no information is available a priori a non-informative Bayesian prior is used.

Converting information from the drawings to a prior occupancy map requires assigning low probabilities to grid cells corresponding to occupied space in the drawing and high probabilities to cells corresponding to empty space in the drawing. A crude prior map could be produced by assigning a high probability to all dark pixels in the drawing and a low probability to white pixels. This however

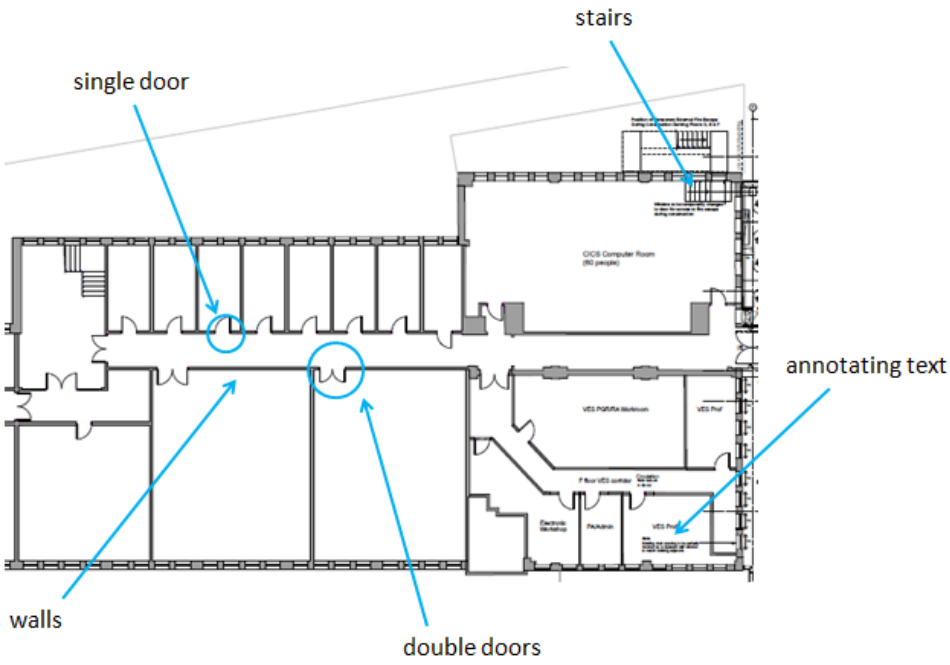


Figure 3.3: An example of an architectural drawing (this is the drawing for one of the Engineering buildings at the University of Sheffield) containing stairs, doors, annotating text and dimension lines.

can be misleading since not all dark pixels correspond to occupied space and not all white pixels correspond to unoccupied space (Figures 3.3 and 3.4). Dark pixels could correspond to annotating text or suggested furniture and white pixels could be part of windows or stairs. Therefore when processing a black-and-white or binarised drawing to extract information that will be used to construct a SLAM prior the aim is to detect dark pixels that correspond to walls.

A number of different types of structural elements are commonly observed in architectural drawings and floor plans. In this thesis these will be referred to as drawing features. Such features include doors, walls, windows, stairs and in-built furniture. Each type of feature can be represented in many different ways depending on the type of drawing, the software used to draw it, the level of detail included, the architect's style and preferences and the type of feature (for example foldable, sliding and double doors are all represented differently, Figure 3.2).

The features that provide the most information about the building outline and occupied and unoccupied space are walls. Walls are less likely to change than features such as suggested furniture and must be present in all buildings unlike stairs, for example, which are optional. Whilst they contain the most information about the building, however, walls are one of the features with the fewest distinguishing attributes.

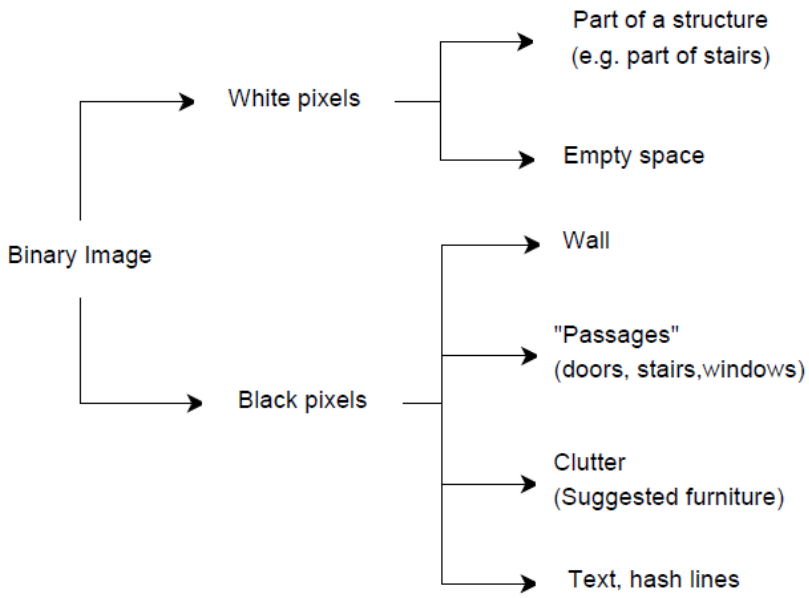


Figure 3.4: Categories of pixels found in a binary image of an architectural drawing-not all dark pixels correspond to occupied space since dark pixels could correspond to annotating text; not all white pixels correspond to empty space since they could be part of a feature such as stairs.

Walls are represented as dark lines in the majority of drawings but have no other distinct geometric features. Trying to detect walls by methods such as searching for thicker lines in the image, for example, does not work for all drawings and floor plans because in some drawings all elements are represented using lines of the same thickness. However, despite the fact that walls are difficult to detect reliably, they have defined relations to other elements that may be easier to detect. For example, doors have distinctive geometric characteristics which can be used to detect them and they need to be attached to a wall. Leveraging this relationship, doors and their relation to walls can be used to deduce wall locations.

Doors are one of the most distinct features found in an architectural drawing. There are a number of different symbols used depending on the type of door represented (double, foldable and sliding are a few examples). Minor differences may also exist between representations used by different architectural firms. A number of door symbols may be seen in Figure 3.5(a) [17]. These are commonly recognised in the literature by using the Hough transform to find the arc that is drawn for most door types [9, 118]. This approach does not generalise well for all types of doors because it does not account for those that do not include an arc (such as foldable doors) and it can also perform quite poorly for low quality images where a clear arc is not visible (Figure 3.6).

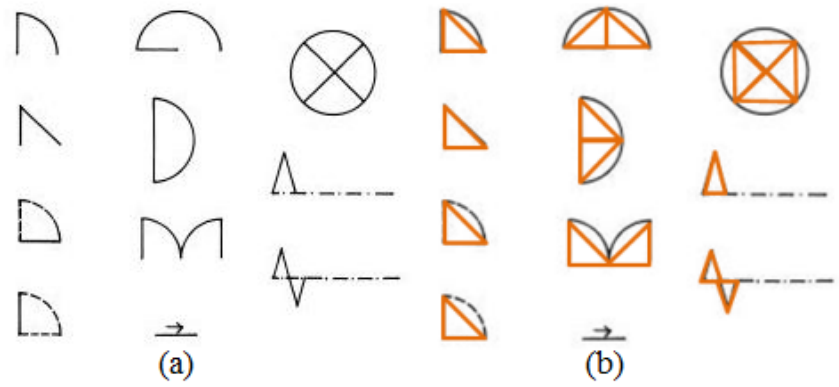


Figure 3.5: (a) Symbols for different types of doors (reproduced from [17]) (b) The majority of door types can be approximated as isosceles triangles (drawn in orange).

Instead of using a template matching approach such as the generalised Hough transform [23], the proposed algorithm uses the geometric characteristics of the door symbols to facilitate drawing independent detection. The key idea behind this algorithm is that all doors can be approximated by isosceles triangles defined by three vertices and (usually) two edges/connected sides (Figure 3.5(b)). The proposed detection strategy searches for triangles in the image (as defined by their vertices) that fulfill the isosceles triangle criteria (at least two equal sides or at least two equal angles).

The advantages of this method are:

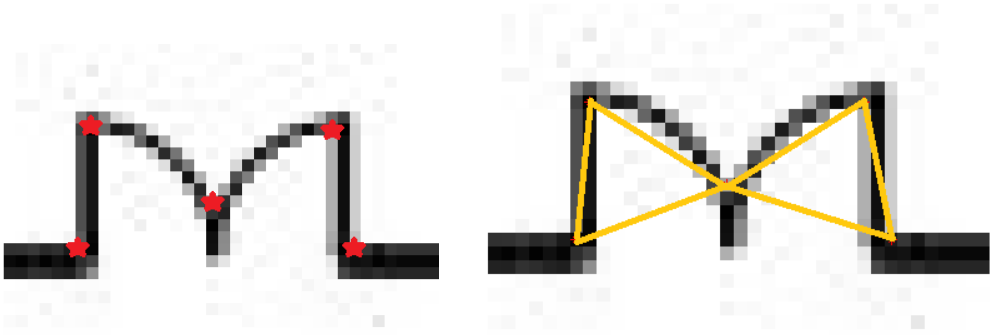
- It generalises well for different drawings, door representations and door types
- It avoids using complex template matching or shape detecting methods that can perform poorly for low quality images.

In order to find vertices of potential triangles the Harris Corners algorithm [70] is used to locate corners in the image. For each pixel in the image this algorithm calculates the change in intensity in a given direction

$$I_n^2 = \frac{n^T \nabla I \nabla I^T n}{n^T n} \tag{3.1}$$

where n is the direction considered and $I(x,y)$ is the intensity of the pixel at location (x,y). I_n^2 is then smoothed by convolution with a Gaussian kernel giving $C_n(x,y)$

$$C_n(x,y) = \frac{n^T A n}{n^T n} \tag{3.2}$$



(a) (b)

Figure 3.6: An example of corners not detected very accurately for a lower resolution image: some of the corners are misaligned which will lead to an inaccurate fitted triangle: (a) Corners detected using the Harris corner algorithm shown as red stars, (b) Triangle fitted to detected corners/vertices shown in orange.

where A are the smoothed values of $\nabla I \nabla I$. Using eigenvalue theory the upper and lower bound of $C_n(x, y)$ are given by the eigenvalues of A. If A has two large distinct eigenvalues then a corner is detected [70].

These vertices detected using Harris Corners are then processed to find all unique three corner combinations that make up a shortlist of potential triangles and hence potential doors. The following tests are then performed on each potential door in the shortlist:

1. Check geometry: door side lengths and angles need to fulfill the isosceles triangle requirement

2. Check connectivity: doors need to have two pairs of connected vertices (a pair of vertices is considered connected if starting from one vertex the other can be reached following only dark pixels)

Geometry test

A potential triangle is defined as $\triangle ABC$, with vertices A, B, C and sides AB, BC, CA. For a potential triangle to be isosceles at least one set of sides must be equal (this could be equivalently tested by checking whether there is at least one pair of equal angles)

$$\{AB = BC \cup BC = CA \cup AB = CA\} \neq \varnothing \tag{3.3}$$

Due to poor image quality the location of detected corners/potential vertices is not always accurate, as shown in Figure 3.6. Therefore this condition is relaxed to testing for at least two approximately equal sides by rounding up calculated

triangle side lengths before searching for equal sides

$$\{AB \approx BC \cup BC \approx CA \cup AB \approx CA\} \neq \varnothing \tag{3.4}$$

After rounding, if no equal sides are found one final test is performed to look for possible isosceles triangles in a very low quality image. If the ratio of any two sides is greater than 0.9 then these are considered roughly equal and the potential triangle passes the sides test

$$\left\{ \frac{AB}{BC} > 0.9 \cup \frac{BC}{CA} > 0.9 \cup \frac{AB}{CA} > 0.9, \frac{BC}{AB} > 0.9 \cup \frac{CA}{BC} > 0.9 \cup \frac{CA}{AB} > 0.9 \right\} \neq \varnothing \tag{3.5}$$

The value of 0.9 was empirically determined to yield good results. Values greater than 0.9 were found to result in virtually the same shortlist as the test in Equation 3.4. Values smaller than 0.9 were found to relax the equal sides condition too much, resulting in a large number of false positives.

The largest out of the three triangle angles, $\angle ABC$, $\angle BCA$ and $\angle CAB$, $angle_{max}$, is also tested. If it is too large then the three vertices may correspond to three collinear points, if it is two small then two of the vertices may be so close they are effectively the same point. If the angle is very large or very small the approximated triangle is also unlikely to correspond to a door. The largest triangle angle is bounded between $60 \leq angle_{max} \leq 100$ since empirically, based on the commonly used door symbols, Figure 3.5, that is an appropriate range for commonly observed door types. Increasing the upper bound leads to three point combinations that are nearly collinear -and thus do not correspond to potential doors- being shortlisted; decreasing the lower limit leads to triangles that have two vertices that are very close to each other being shortlisted which are also unlikely to correspond to doors.

Connectivity test

After the geometric tests are completed a connectivity test is performed on the shortlisted potential doors. In order to pass this test at least two sets of the vertices of the approximated triangle $\triangle ABC$ need to be connected. Two vertices are considered to be connected if starting at one of them the other can be reached following consecutive black pixels. The details of the connectivity check for two vertices are as follows. A vertex is chosen as the starting point with coordinates (x_o, y_o). The aim is then to attempt to reach a destination vertex, (x_d, y_d) by attempting to follow only dark pixels. Starting at (x_o, y_o) neighbouring pixels are

tested. If one dark pixel is found it is chosen as the next pixel to move to and if more than one dark pixels are found a further test is performed to determine which is closest to (x_d, y_d). If the destination is reached by following such dark pixels then the origin/destination vertex set passes the connectivity test.

The relative position of (x_o, y_o) and (x_d, y_d) is used to determine which neighbouring pixels to test at each iteration. A different set of pixels is tested depending on whether the destination pixel, (x_d, y_d) is to the left or right of (x_o, y_o) and whether it is lower or higher in the image than (x_d, y_d). The origin of the coordinate system used in the image is set to be the topmost leftmost point, signifying that pixels to the right of the image have larger x coordinate values than those further left and pixels higher in the image have lower y values than those towards the bottom of the image. Depending on the location of the destination vertex relative to the origin vertex different neighbouring pixels are tested.

If the destination vertex, (x_d, y_d), is further up and further right than (x_o, y_o) the coordinates of the next three pixels to test are given by

$$P_n = \{(x_c + 1, y_c), (x_c, y_c - 1), (x_c + 1, y_c - 1)\} \quad if \quad \{x_o < x_d \cap y_o > y_d\} \tag{3.6}$$

If the destination vertex, (x_d, y_d), is further up and further left than (x_o, y_o) the coordinates of the next three pixels to test are given by

$$P_n = \{(x_c - 1, y_c), (x_c, y_c - 1), (x_c - 1, y_c - 1)\} \quad if \quad \{x_o > x_d \cap y_o > y_d\} \tag{3.7}$$

If the destination vertex, (x_d, y_d), is further down and further right than (x_o, y_o) the coordinates of the next three pixels to test are given by

$$P_n = \{(x_c + 1, y_c), (x_c, y_c + 1), (x_c + 1, y_c + 1)\} \quad if \quad \{x_o < x_d \cap y_o < y_d\} \tag{3.8}$$

Finally, if the destination vertex, (x_d, y_d), is further down and further left than (x_o, y_o) the coordinates of the next three pixels to test are given by

$$P_n = \{(x_c - 1, y_c), (x_c, y_c + 1), (x_c - 1, y_c + 1)\} \quad if \quad \{x_o > x_d \cap y_o < y_d\} \tag{3.9}$$

(x_c, y_c) are the coordinates of the current pixel and P_n is the set of neighbouring pixels to be examined. These are determined depending on the relative location of the destination to the origin as described mathematically in Equations 3.6, 3.7, 3.8 and 3.9 and visualised in Figure 3.7. These neighbouring points P_n are chosen to ensure that the direction of movement is the direction of the target (the target could be reached through a long convoluted route but that would not give information about the door of interest).

Algorithm 3.1 Connectivity check

$(x_c, y_c) \leftarrow (x_o, y_o)$
$I((x_c, y_c)) \leftarrow$ dark
DestinationReached $\leftarrow$ no
Connected$\leftarrow$ yes
while DestinationReached == no AND $I((x_c, y_c))$ == dark AND Connected$\leftarrow$ yes **do**
  $P_n \leftarrow$ FindNeighbours
  $n_d \leftarrow$ FindDark(P_n)
  if $n_d == 0$ **then** Connected$\leftarrow$ no
  else if $n_d == 1$ **then** $(x_c, y_c) \leftarrow (x_N^{dark}, y_N^{dark})$
  else
    for i=1:3 **do**
      $d_i \leftarrow \sqrt{(x_N^i - x_{dt})^2 + (y_N^i - y_d)^2}$
    end for
    $(x_c, y_c) \leftarrow (x_N^{min(d_i)}, y_N^{min(d_i)})$
  end if
end while

In order to select which of the shortlisted three neighbouring pixels P_n to move to next the intensities of the three neighbouring pixels, $I(P_n)$, are examined to determine how many out of the neighbours P_n are dark:

- If $n_d = 0$: the two vertices fail the connectivity test
- If $n_d = 1$: the dark pixel is chosen as the new current pixel , (x_c, y_c)
- If $n_d > 1$: further testing is required

where $0 \leq n_d \leq 3$ is the number of dark pixels out of the set P_n.

If $n_d > 1$ then the next point to move to out of the possible P_n is determined by testing which of the points in P_n is closest to the destination vertex (x_d, y_d). The Euclidean distance between each of the neighbouring pixels (x_N^i, y_N^i), $i = 0, ..., 3$ and the destination vertex (x_d, y_d)

$$d_i = \sqrt{(x_N^i - x_d)^2 + (y_N^i - y_d)^2} \tag{3.10}$$

is used to determine which of the three neighbouring pixels will be the pixel to move to and the pixel (x_N^i, y_N^i) that is the closest to the destination vertex, the pixel with the smallest d_i, is chosen. The overall algorithm used to perform the connectivity test is shown in Algorithm 3.1.

False positives removal

After the connectivity tests are completed the shortlist of doors is passed through two final tests to remove any false positives detected in feature rich areas such as

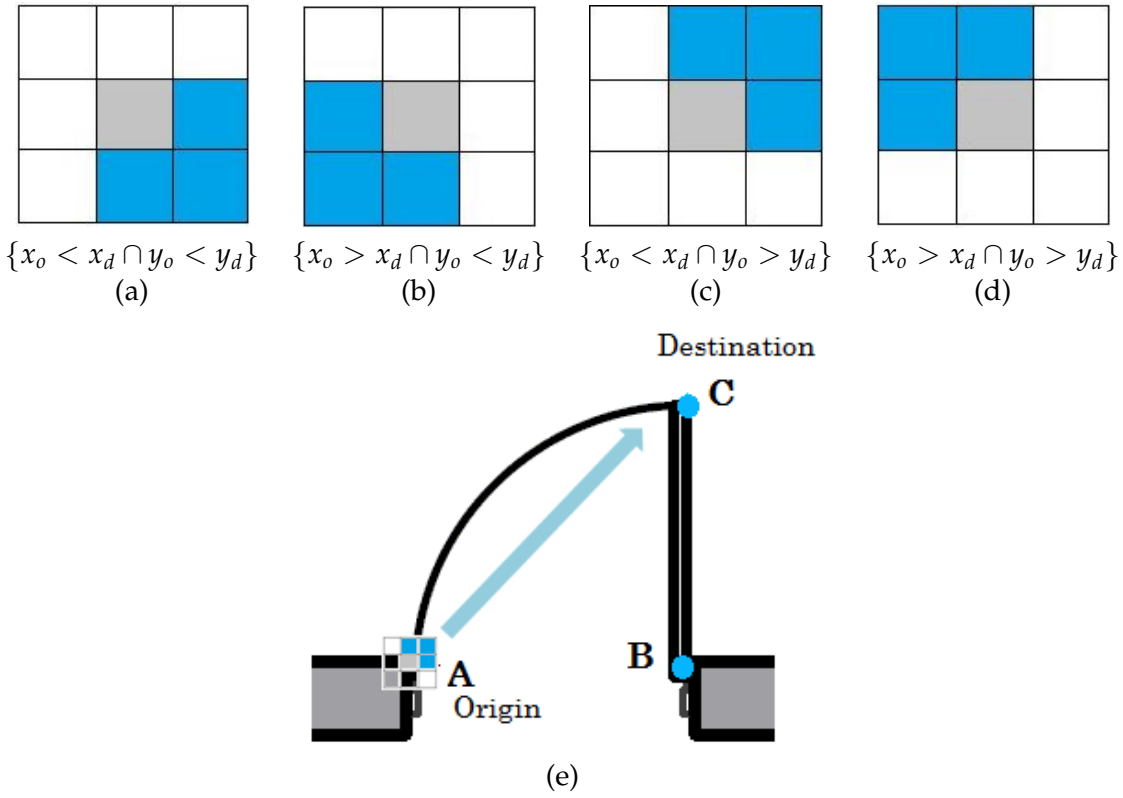


Figure 3.7: Choosing pixels to move to starting at grey pixel, (x_c, y_c): depending on destination vertex location the pixels to examine are highlighted in blue: (a) down and right, (b) down and left, (c) up and right and d) up and left; (e) shows an example of an origin and destination pair.

stairs:

1. Check intensity of pixels within the triangle: the percentage of dark pixels needs to be smaller than 40% (this value was tested for drawings with very different representations and was found to yield good results)

2. Remove repeated/overlapping doors: If the areas of the approximated triangles overlap more than 30% (this value was deduced empirically) then only one of them is maintained

3. Check no door points are on the outer walls: common false positives such as stairs tend to have vertices on outer walls; outer walls are detected separately and hence no information is lost by discarding such potential doors

The thresholds for the above checks were determined as follows. When checking the intensity of the pixels within the triangle the aim is to remove triangles that contain many black pixels and could correspond to false positives at feature-rich areas such as stairs. The threshold should thus be a small number, ideally. Setting a low value such as 10% was found to yield many false negatives, with true doors being missed. This was due to corners not being detected at the exact locations of vertices, especially in the case of lower quality images. This resulted in fitted triangles that contain some dark pixels in the fitted triangle. This number was thus increased until a value was found that avoided missing many true doors but also did not result in many false positives.

A number of values were tested to determine an appropriate threshold for the acceptable percentage overlap between triangles. The threshold should ideally be low- a percentage of overlap of 50% would indicate two triangles effectively corresponding to the same door. In practice, this value was found to be high and result in many duplicates not being detected and was gradually decreased until a threshold that removed the majority of duplicates for all images tested was found, 30%.

In order to detect the outer walls the Harris corners closest to the image boundaries are first shortlisted

$$C_{edge} = \left\{ (x^C_{min}, y^C_{min}),\ \ (x^C_{min}, y^C_{max}),\ \ (x^C_{max}, y^C_{min}),\ \ (x^C_{max}, y^C_{max}) \right\} \tag{3.11}$$

where x^C_{min} is the x coordinate of the leftmost detected corner, x^C_{max} that of the rightmost detected corner and y^C_{min}, y^C_{max} are the lowest and highest y coordinates of detected corners.

The intensities of the pixels along the horizontal and vertical directions are then tested to find dark pixels along those lines. These are then stored as outer wall points

$$Walls_{out} = (x, y) \in \left\{ H_{points} \cup V_{points} \right\} \tag{3.12}$$

with

$$H_{points} = \left\{ (x, y) | x^C_{min} \leq x \leq x^C_{max},\ y = \left\{ y^C_{min},\ y^C_{max} \right\},\ I(x, y) = 0 \right\} \tag{3.13}$$

$$V_{points} = \left\{ (x, y) | y^C_{min} \leq y \leq y^C_{max},\ x = \left\{ x^C_{min},\ x^C_{max} \right\},\ I(x, y) = 0 \right\} \tag{3.14}$$

where H_{points} are the points along the horizontal direction and V_{points} those along the vertical direction and $I(x, y)$ the intensity of a pixel with coordinates (x, y), with $I(x, y) = 0$ identifying dark pixels in a binarised image.

This outer wall detection is performed to remove false positives; the detected outer walls are amended when walls in the image are detected as described in the next section.

Wall detection

The next step after determining the final door shortlist is to detect walls attached to the doors to construct the desired occupancy grid prior. Walls are the main feature needed to construct an occupancy grid map prior since they give the outline of the building and must be present for all buildings.

First of all, the outer walls are detected as described in the previous section and added to the list of wall points, W. Thus W is intialised as $Walls_{out}$, Equation 3.12. Each detected door is then examined to find walls attached to it.

For each door the pair of vertices that failed the connectivity test (Agorithm 3.1), (white door side, AB in Figure 3.8) are used. Each shortlisted door has two pairs of connected vertices. The vertices that are not connected are the ones that are attached to walls, Figure 3.8. Using these two vertices a similar method to the one used to find outer walls is used to find wall points.

The aim of the algorithm is to start from vertex V_o (A in the case of Figure 3.8) and keep following lines of dark pixels perpendicular or parallel to AB until a white pixel is found/the end of a wall is reached.

The orientation of the door symbol is first determined by checking whether the unconnected vertices are aligned horizontally or vertically

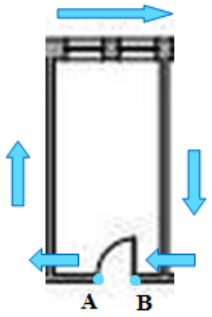


Figure 3.8: Find walls by starting at vertex A and finding the first dark pixel along AB then follow all dark pixels along that line until a white pixel is reached. Then check directions perpendicular to AB until no more dark pixels are found, then pixels perpendicular to that direction and so on until vertex B or the end of the image is reached.

$$Orientation = \begin{cases} h & if \ \mid x_{V_d} - x_{V_o} \mid > \mid y_{V_d} - y_{V_o} \mid \\ v & if \ \mid x_{V_d} - x_{V_o} \mid < \mid y_{V_d} - y_{V_o} \mid \end{cases} \tag{3.15}$$

In order to determine whether the door's unconnected side is vertical with $Orientation{=}v$, or horizontal with $Orientation{=}h$, the difference between the x and y coordinates of each origin and destination pair are compared. The vertices are unlikely to be perfectly aligned vertically or horizontally and so the absolute difference between the x coordinates is compared to the absolute difference between the y coordinates of the origin and destination vertices. If the discrepancy in x coordinates is greater than that in y coordinates the door orientation is horizontal, h, and if the opposite is true it is vertical, v, Equation 3.15. If the absolute differences are found to be equal the door is not used to find walls. This is because this system assumes that doors are either vertically or horizontally aligned (or nearly vertically or horizontally aligned). Doors at an angle are less common and they are deemed to be beyond the scope of this thesis.

After the orientation of the door is determined, pixels parallel and perpendicular to the segment $\overline{V_o V_d}$, corresponding to AB in Figure 3.8, are examined and dark pixels along those directions are stored as wall points.

In order to explain how the wall detection algorithm works, the origin vertex $V_o = A$ in the horizontal door of Figure 3.8 will be used. Starting at V_o and because $Orientation = h$ pixels along the line parallel to AB are examined. Only pixels with $0 < x_i < x_{V_o}$ and $0 \leq y_i \leq y_{max}$ are tested, namely pixels that are to the left of A and within the image boundaries, with y_{max} being the image y dimension. The x bounds are chosen because $x_A < x_B$ in this case so there will be

Algorithm 3.2 Drawing Processing Algorithm

```
Find Harris corners
Find all unique three corner combinations
for all potential doors do
    if side test successful AND connectivity test successful then
        save door
    end if
end for
Detect outer walls
Remove doors on outer walls
Remove doors with dark points in triangle
Find walls
```

no wall points for $x_A < x < x_B$.

In a horizontal sweep the following points are added to the detected walls, W

$$\{(x,y) \in ([0,x_{V_s}),[0,y_{max}]), I(x,y)=0, I(x-1,y)=0, I(x,y-1)=0\}\,,\ x<x_{V_s} \quad (3.16)$$

$$\{(x,y) \in ((x_{V_s},x_{max}],[0,y_{max}]), I(x,y)=0,\ I(x-1,y)=0, I(x,y-1)=0\}\,,\ x>x_{V_s} \quad (3.17)$$

where (x_{max} and y_{max}) are the image x and y dimensions, x_{V_s} the starting point of the sweep and $I(x,y)$ the intensity of the pixel at location (x,y). $I(x-1,y)=0$ means that the previous horizontal pixel was black and $I(x,y-1)=0$ means that the previous vertical pixel was black to ensure we stop saving wall points once a white section is reached).

Similarly in a vertical sweep

$$\{(x,y) \in ([0,x_{max}],[0,y_{V_s})), I(x,y)=0, I(x-1,y)=0, I(x,y-1)=0\}\,,\ y<y_{V_s} \quad (3.18)$$

$$\{(x,y) \in ([0,x_{max}],(y_{V_s},y_{max}]), I(x,y)=0, I(x-1,y)=0, I(x,y-1)=0\}\,,\ y>y_{V_s} \quad (3.19)$$

The horizontal and vertical sweeps are performed in succession, starting with the appropriate sweep depending on door orientation and performing a sweep in each direction twice. The test is performed twice to avoid constantly detecting the same walls: two sweeps in each direction should be enough to detect the room the door is attached to.

The overall algorithm used to process the drawings is shown in Algorithm 3.2 and each algorithm step is shown in Figure 3.9. Detected walls can be translated directly to the occupancy grid map format by assigning a low probability of being unoccupied to all detected wall points. This algorithm has a computational cost of $O(n^3)$ where n is the number of Harris Corners detected in the image and it is dominated by the cost of finding the possible combinations $nCr = \frac{n!}{r(n-r)!}$ of $r=3$ out of n detected Harris corners.

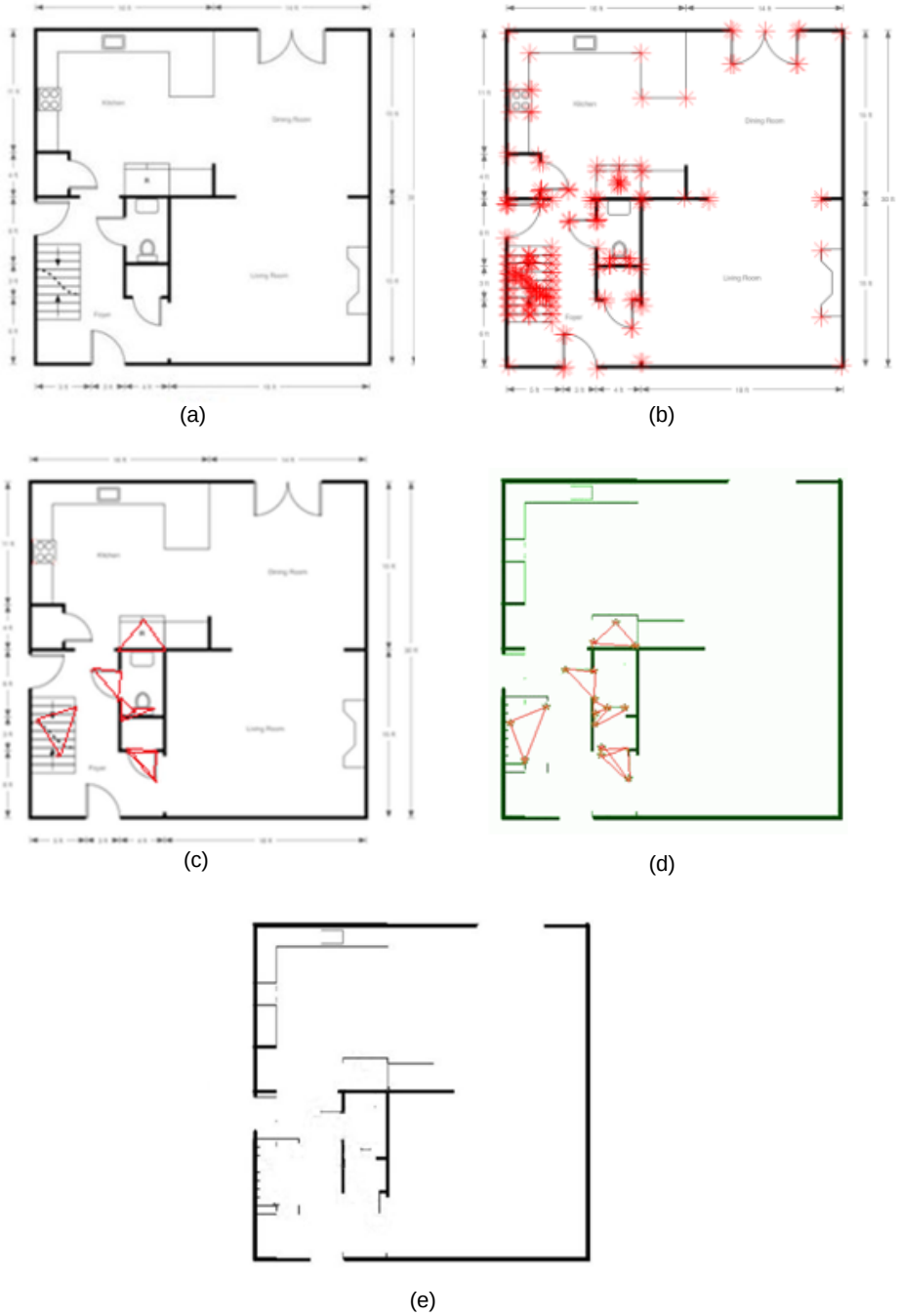


Figure 3.9: The proposed algorithm to detect walls in a floor plan or architectural drawing. (a) Input drawing; (b) Corners detected using the Harris Corners algorithm, setting the number of corners to be detected to 200; (c) Shortlisted doors; (d) Walls detected (green lines) following the shortlisted doors (orange triangles), with most walls being identified correctly despite some false positive doors; (e) Algorithm output.

3.3.2 Assessing drawing quality

In order to assess the performance of the drawing processing algorithm presented in Section 3.3.1 a quantitative method to assess the quality of the extracted information is required. The quality of extracted information is often assessed by calculating the percentage of features detected correctly or by using similar metrics originally designed to assess the quality of recognised symbols in mechanical drawings [126]. Such approaches, however, are a poor way of determining how suitable a constructed prior map is since they focus on per-symbol recognition accuracy and not accuracy of wall detection, which is the focus of this chapter.

A set of criteria to assess the suitability of extracted structural information to construct an occupancy prior are proposed as follows

1. Correctly detecting walls is the main system requirement and hence should be the primary performance metric
2. Missing a wall is less of a problem than detecting a false positive since a SLAM system can fill-in missing walls whereas false positives may restrict areas the robot is deployed in
3. Detecting stairs or in-built furniture as walls should not be penalised: these are strictly not wall locations but they behave similarly since they cannot be traversed or moved

Given the above criteria a novel approach is proposed to assess the quality of the extracted information: the recognition is formulated as a classification problem. Each pixel in the image can be classified as belonging to one of the categories seen in Figure 3.4. This way of assessing performance can be mapped to the aims of constructing occupancy grid priors, separating regions in the grid as occupied or unoccupied. The algorithm proposed in Section 3.3 focuses on wall detection and so the generalised problem formulation is converted to a binary classification problem. This simplification provides an assessment of how successful the system is in terms of correctly classifying pixels as being occupied or unoccupied, which is the main performance criterion for accurate prior construction. This approach can be extended and used for other systems that aim to detect other features in the image such as stairs for example.

Viewing the proposed recognition algorithm as an algorithm to detect walls in the image, clutter and text are considered to be true negatives; walls and windows are considered to be true positives and doors, stairs and in-built furniture are considered to be a "don't care state" (the system is not penalised for detecting them as positives or negatives) as shown in Figure 3.10. Doors, stairs and inbuilt

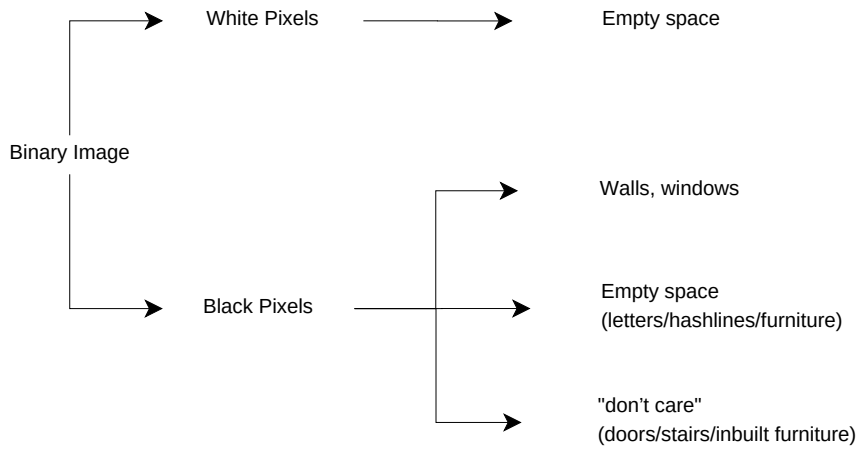


Figure 3.10: Simplified classification problem: white points in the image, clutter and text are considered to be true negatives; walls and windows are considered to be true positives and doors, stairs and in-built furniture are considered to be a "don't care state" (the system is not penalised for detecting them as positives or negatives).

furniture may be classified as occupied or unoccupied space and neither of these decisions will be deemed a false positive. Classifying them as occupied is strictly not false since they represent obstacles in the real world or areas a robot could not access but, conversely, they are not true walls. If the original algorithm is extended to include clutter and stair detection the classification evaluation can be expanded to include more possible classes but this is outside the scope of this thesis.

The extracted walls and empty space image contains the following information. Set Im represents the entire image, set TW the true walls and set DW the detected walls. Set $Im - TW$ represents the actual negatives, $TW \bigcap DW$ represents the true positives, $TW - TW \bigcap DW$ the false negatives and $DW - TW \bigcap DW$ the false positives in the image. Three metrics commonly used for classification evaluation are used to assess performance [39]:

- $Precision = \dfrac{TruePositives}{WallsDetected} = \dfrac{TW \bigcap DW}{DW}$
- $Recall(Sensitivity) = \dfrac{TruePositives}{ActualWalls} = \dfrac{TW \bigcap DW}{TW}$
- $False\ positive\ rate = \dfrac{FalsePositives}{ActualNegatives} = \dfrac{DW - TW \bigcap DW}{Im - TW}$

Precision calculates the percentage of pixels correctly identified as walls out of all the pixels detected as walls. Recall is the percentage of wall pixels identified correctly out of all walls in the image. The false positive rate signifies the percent-

age of misclassified empty space pixels out of all pixels corresponding to empty space.

3.4 Results and discussion

The algorithm presented in Section 3.3.1 was used to process representative drawings and the performance assessment method proposed in Section 3.3.2 was used to assess performance and qualitative and quantitative results are presented in this section.

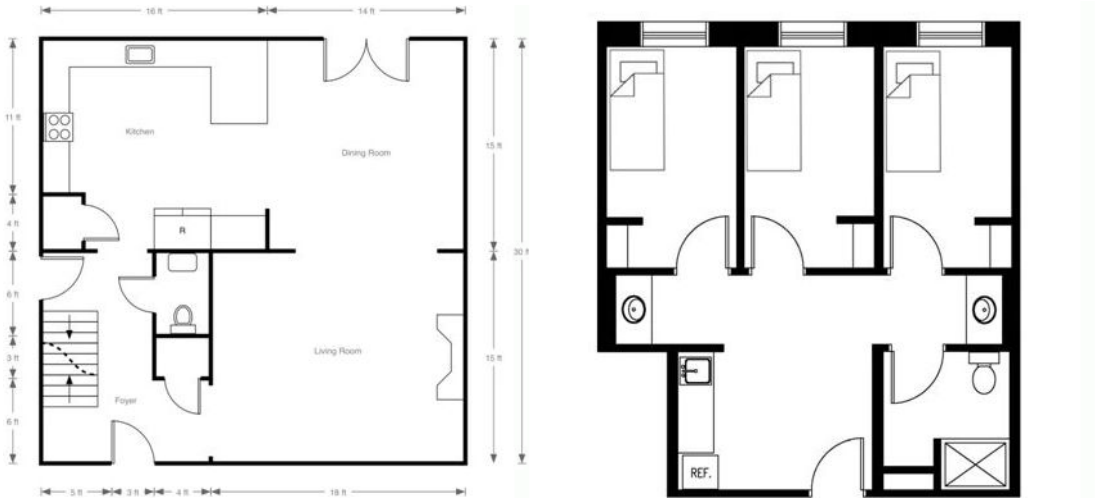
3.4.1 Image test set selection

Five representative images were selected to assess performance. Two floor plans were selected, one with dimension labelling and inbuilt furniture in light grey (Figure 3.11(a)) and a floor plan with multiple suggested furniture and details (Figure 3.11(b)). Three sections of an architectural drawing of a University of Sheffield Engineering building were also selected. Figure 3.12(a) contains stairs which are a common source of false positives as well as double doors which are more challenging to detect. Figure 3.12(b) presents a relatively straightforward drawing section and Figure 3.12(c) is most challenging containing stairs, double doors, labelling and a door at an angle. This set of drawings was chosen to test the performance of the algorithm since it covers a wide range of symbols, representations and formats.

3.4.2 Qualitative drawing processing results

Results produced for drawings discussed in the previous section are shown in Figures 3.13 and 3.14. The proposed algorithm performs well for drawings with few features such as doors and not much clutter, Figure 3.14 (a) and (b). Performance is worse for Figure 3.14 (c) because there are many more features but also some of the doors are more difficult to detect (such as the door that is at an angle). The detected walls (seen in green, middle column of Figures 3.13 and 3.14) and the algorithm's output, the extracted walls (rightmost columns Figures 3.13 and 3.14), are accurate despite occasionally detecting some false positive doors or missing some of them. However because of the way the algorithm works more accurate door detection does lead to more accurate wall detection.

A point to consider is that the number of detected doors depends on the corners detected in the image. For images where walls are represented with thicker



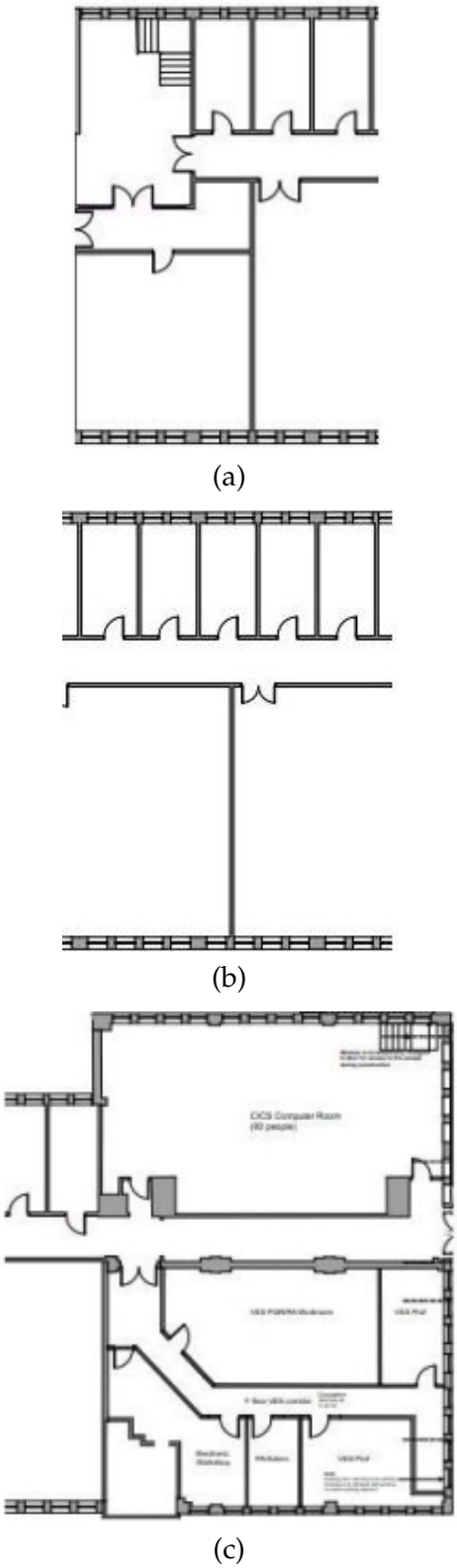
(a) (b)

Figure 3.11: Image test set floor plans. Drawing (a) contains labelling, thin and thick lines, stairs, double doors and inbuilt furniture, drawing (b) contains suggested furniture and details.

lines than doors all corners may be detected at wall intersections since such corners will appear stronger [70]. In order to avoid this problem the input drawing can be segmented into smaller sub-images containing fewer features to ensure no doors are missed. A divide and conquer strategy can then be used to detect doors more accurately in each segmented image and the extracted walls for each sub-image can be collated to produce the final extracted walls. This approach can yield computational savings because a smaller number of Harris corners would need to be detected per sub-image without missing corners around features of interest.

The method of using dilation to differentiate between thinner and thicker lines in the image and thus differentiate between walls and other structural elements [10, 11, 100], as discussed in Section 3.1, was implemented in order to compare its performance to that of the proposed approach. Dilation detects pixels in the boundaries of objects in the image [145]. In order to detect such boundaries the intensity of each pixel i at location (x_i, y_i) and a region of its neighbours, as determined by a structuring element, are compared. The largest intensity value is selected as the dilated value for position (x_i, y_i). The structuring element chosen determines the shape of the region of neighbours that is examined.

The Matlab implementation of Dilation was used, dilating the image using a square as a structuring element object, with two different possible square sides presented in the results. A square was chosen since walls, the elements we aim to detect, are rectangular. Two different square sizes were used to examine the effect



(a)

(b)

(c)

Figure 3.12: Image test set architectural drawing sections of one of the University of Sheffield's Engineering buildings. (a) contains stairs and double doors, (b) is a more straightforward drawing to process and (c) presents a number of difficulties including text, stairs, labelling and a door at an angle.

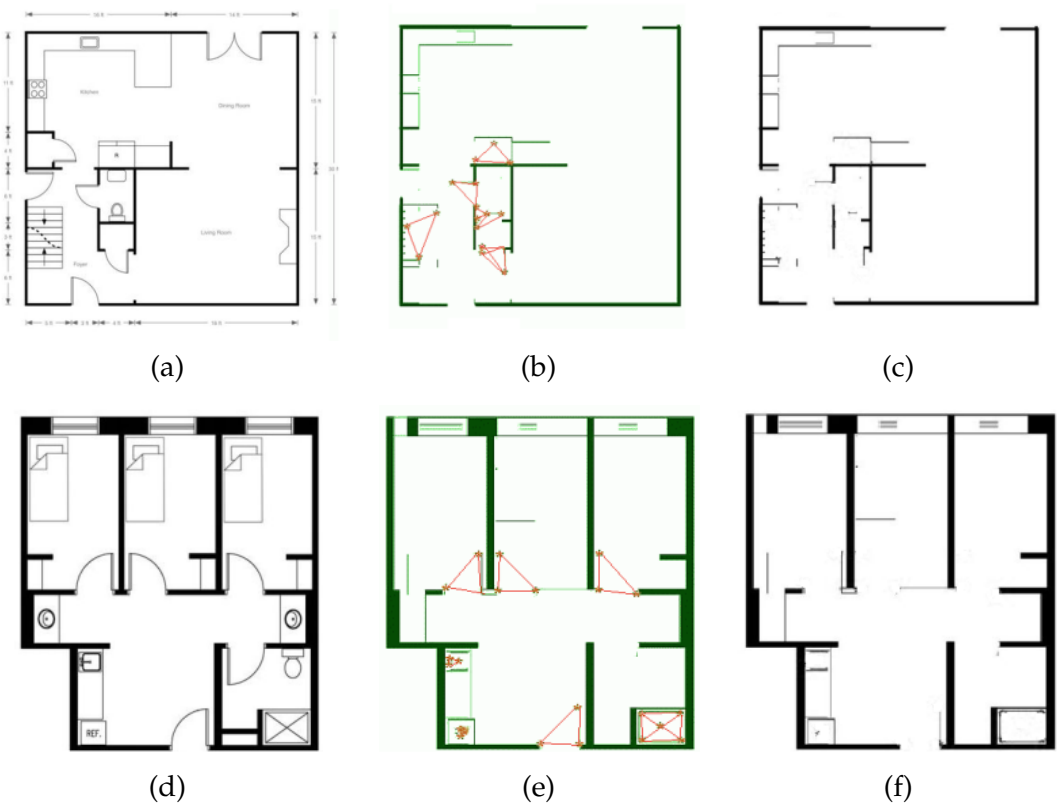


Figure 3.13: Results of drawing processing, with each row showing the process for a different floor plan. (a) and (d) are the input drawings, (b) and (e) show the extracted information, with detected doors shown as orange triangles and detected walls shown in green, (c) and (f) are the algorithm's output for each case.

of the size of each structural element.

The drawings in the left column, (a) and (c) in Figure 3.15 use a square element object of side 2, while the right column, (b) and (d), use a square side of 4. The left column contains too much information, failing to eliminate doors and suggested furniture, especially in the case of the floor plan shown in (c). Conversely, using a larger square side of 4 leads to most information being eliminated in the case of the architectural drawing in (b) and windows being displayed as doors in (d). Neither square side performs well, with smaller sides making virtually no difference and larger sides eliminating information. Moreover there is no square value that performs well for both drawings: floor plans with varying line widths for different elements may respond well to larger square sizes but a larger square size is likely to lead to very few wall points being correctly identified in architectural drawings.

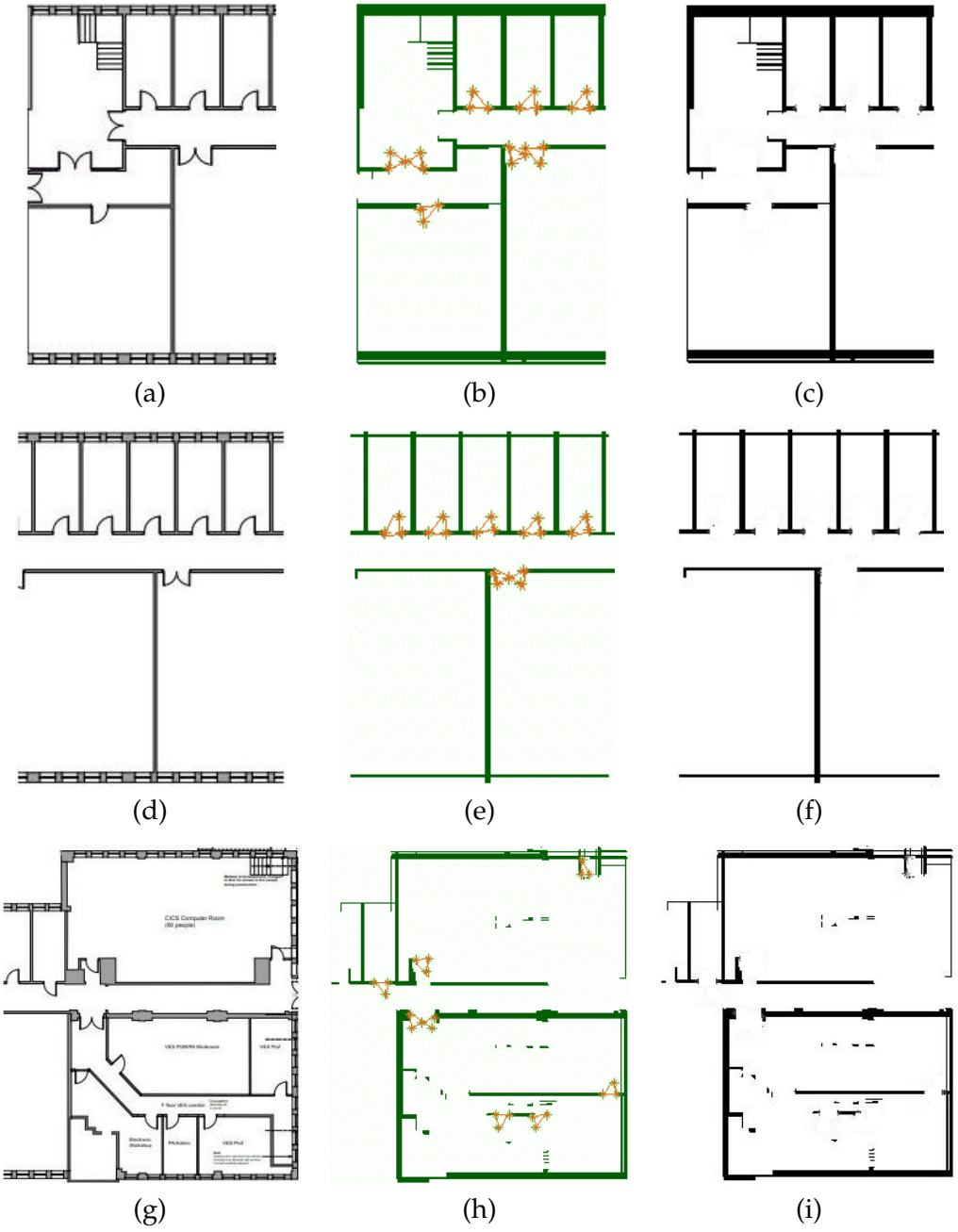
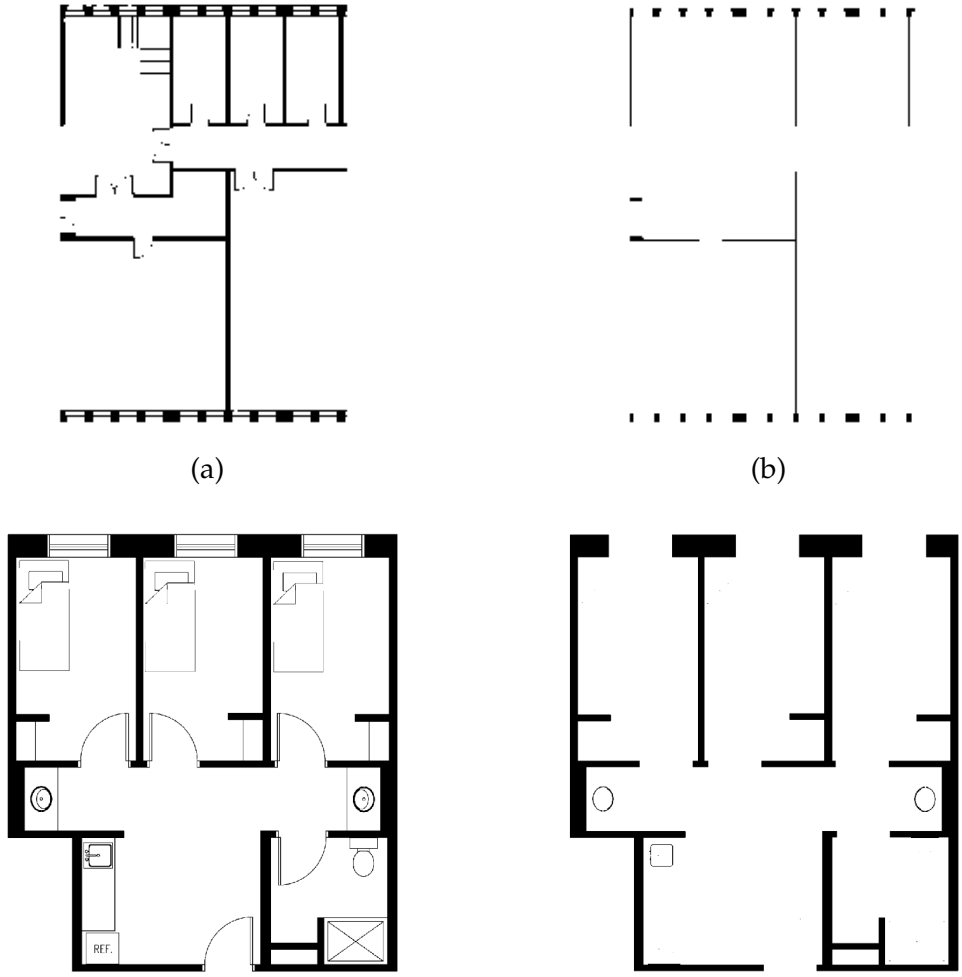


Figure 3.14: Results of drawing processing, with each row showing the process for a different architectural drawing section. (a), (d) and (g) are the input drawings, (b),(e) and (h) show the extracted information, with detected doors shown as orange triangles and detected walls shown in green, (c), (f) and (i) are the algorithm's output for each case.



(a) (b)

(c) square side 2

(d) square side 4

Figure 3.15: Extracted walls using dilation and a square as a structuring element object for an architectural drawing, (a), (b) and for a floor plan, (c), (d). The left column, (a) and (c), use a square element object of side 2, while the right column, (b) and (d), use a square side of 4.

3.4.3 Quantitative assessment

The metrics described in Section 3.3.2 were chosen to give a measure of how accurate the system is in terms of detecting true walls but also the fraction of the actual walls it detects. The ground truth (true walls, true negatives) was extracted by hand and compared to the detected walls. The values of these metrics for each of the drawings presented in the results section are shown in Table 3.1.

Drawing	Precision %	Recall %	False positive rate %
Figure 3.11a	98.97	94.46	0.04
Figure 3.11b	99.23	69.96	0.09
Figure 3.12a	99.51	91.13	0.05
Figure 3.12b	99.88	66.17	0.01
Figure 3.12c	98.70	61.28	0.12

Table 3.1: Precision, recall and false positive rate calculated for the information extracted each of the drawings in Section 3.4.1 using the algorithm proposed in Section 3.3.1.

As can be seen in Table 3.1 drawings for which results are very good in terms of a qualitative analysis such as Figure 3.11(b) and Figure 3.12(b) get quite low scores in terms of recall. That is partly due to the fact that, for drawings with thick walls, some pixels along the wall thickness may not be detected. This does not reduce the quality of the prior constructed since the innermost lines defining the outer walls of a building constrain the environment the robot can move in very accurately.

In terms of precision the system performs very well for all drawings, so the number of false positive wall points is very low. Finally the false positive rate is very low for all drawings: some walls may be missed but the system manages to make very accurate predictions of wall locations.

The results obtained using the proposed method to detect walls were also compared to the results obtained using dilation as described in the qualitative results. As shown in Figure 3.16, the proposed approach outperforms using dilation for both square sizes used. In some cases using dilation with an appropriate square size can yield improved recall, such as in Figure 3.16(b) and (e) for a square side of 2, or a smaller false positive rate in Figure 3.16(c) for both square sizes and for a square side of 2 for (b) and (e). This however is not true across all drawings, further confirming the fact that this method does not generalise well for all drawings.

These results indicate that the proposed algorithm fulfills the defined applica-

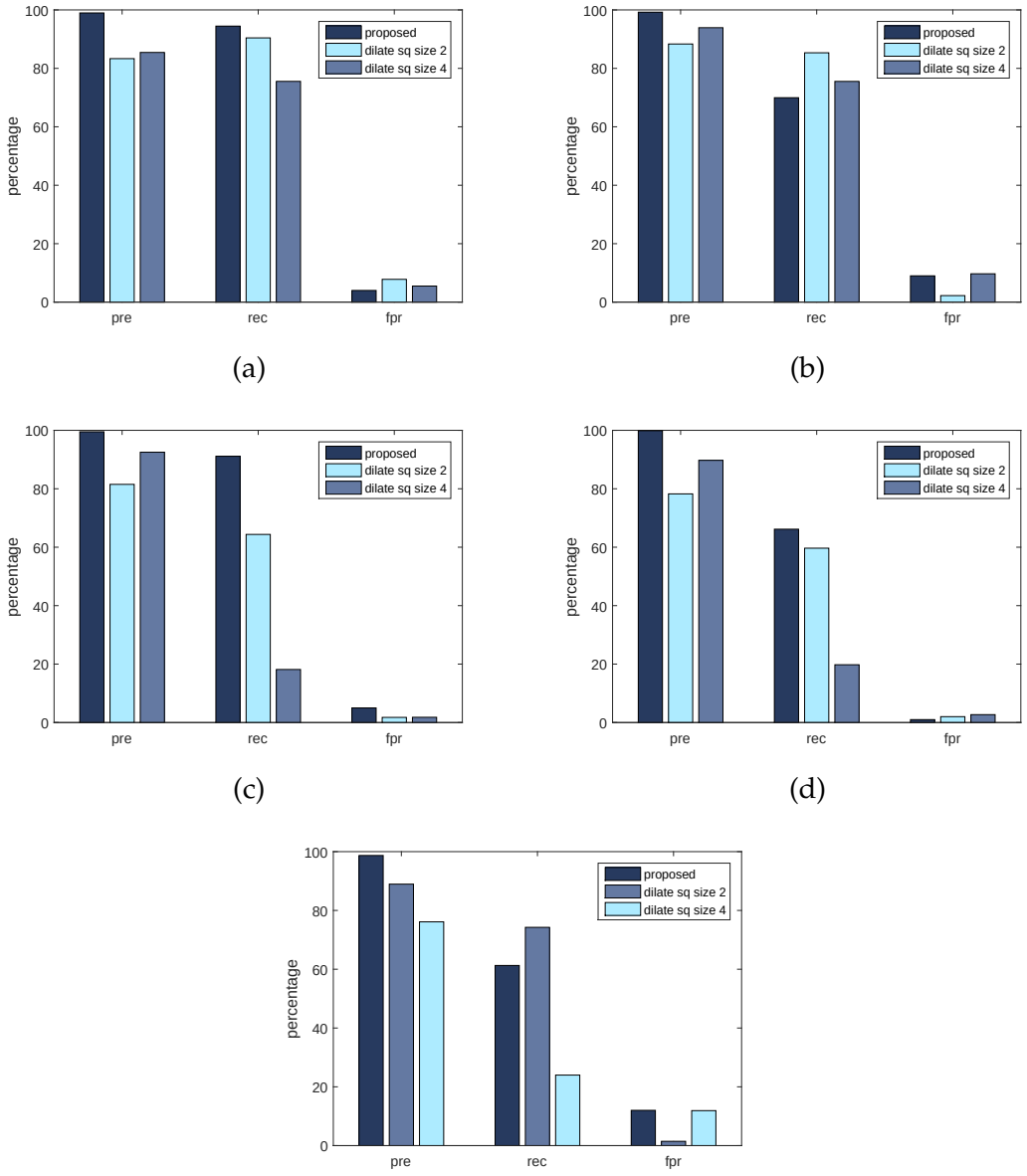


Figure 3.16: A quantitative comparison of the precision, recall and false positive rates of the walls extracted using the proposed approach and the commonly used dilation method for two different square sizes for five different drawings. (a) results for drawing (a), Figure 3.13; (b) results for drawing (b), Figure 3.13; (c) results for drawing (a), Figure 3.14; (d) results for drawing (b), Figure 3.14; (e) results for drawing (g), Figure 3.14.

tion specific criteria well, yielding a high precision for all drawings tested, meaning very few of the detected wall pixels are false positives.

3.5 Concluding remarks

A novel system to process floor plans and architectural drawings and extract SLAM prior information using a set of geometric constraints and tests was proposed in this chapter. Unlike existing methods the proposed algorithm prioritises wall detection, since walls are structural elements that are present in all buildings and determine the building outline. A set of criteria and metrics to assess the quality of the extracted prior information was also proposed, viewing the problem of wall detection as a binary classification of image pixels as belonging to either walls or empty space. Current performance is good for drawings that do not contain many features and the set criteria are met, with tested drawings yielding a precision of over 98%, recall of over 61% and false positive rates lower than 0.09%. The extracted wall locations can be used by a robot to localise in the indoors environment. In order to be used as a mapping prior, however, they need to be processed and placed into a Bayesian prior format as discussed in the next chapter.

Chapter 4

Identifying optimised prior values

Improving SLAM performance by constructing and using meaningful priors can be very beneficial for time and safety critical applications such as USAR missions. Using an optimal prior can yield improved performance without increasing the computational complexity of the SLAM algorithm itself. There is only a one-off cost of $O(n^3)$ associated with extracting appropriate structural information and constructing a prior map, but that can be performed offline.

The main contribution of this chapter is a study of possible contextual Bayesian indoors mapping priors. Building on the work presented in Chapter 3, different possible prior values are tested and maps produced are compared to those produced using a generic, non-informative prior that assigns equal prior probabilities to all locations (as is typically done in the literature). Prior values for detected walls and empty space in the architectural drawing or floor plan that optimise precision and recall are thus identified and optimised informative priors are constructed. The work presented in this Chapter is published in the International Journal of Robotics Research [60].

The contributions of this chapter are as follows:

- A new occupancy grid mapping simulator for Matlab that allows the study of the effects of different prior maps
- A study of how prior values of occupancy assigned to detected walls and empty space affect map quality
- Identifying optimised prior values to assign to empty and occupied space locations detected in the architectural drawing

4.1 Background

SLAM aims to have a robot equipped with sensors explore and map an environment whilst simultaneously localising itself within it. The robot uses sensors to perceive the environment, a motion model or odometer to predict robot motion and it can also use a prior to incorporate information about the environment [50]. Numerous solutions to the SLAM problem have been proposed over the years [47, 50] each trying to improve performance by improving a different aspect. However, very little research has been conducted in the area of constructing and using Bayesian priors to improve performance.

This chapter addresses this problem by proposing a method to construct contextual priors that improve the performance of SLAM without increasing computational complexity. It explores this concept by presenting a case study for indoors SLAM optimised prior construction, Figure 4.1.

First the probabilistic formulation of SLAM is given, highlighting the role of the prior map. The implementation of choice, FastSLAM [109], is then presented, showing the separation of the localisation and mapping tasks using Rao-Blackwellization. This separation allows a study of mapping without the need to focus on performing localisation. Occupancy grid FastSLAM is then presented, highlighting the incorporation of prior information through the occupancy grid mapping algorithm. Occupancy grids represent the area to be explored by a grid of cells each associated with a probability of occupancy. They are the map representation chosen in this thesis because the incorporation and visualisation of prior information is relatively straightforward. The map at time $k = 0$ is the Bayesian prior constructed based on available information and if no information is available a priori a non-informative Bayesian prior is used, assigning a prior probability of 0.5 to all grid squares. Information extracted from architectural drawings or floor plans using the method proposed in Chapter 3 is leveraged in this chapter to construct optimised Bayesian priors that can improve map accuracy.

4.1.1 Probabilistic formulation

SLAM performs a Bayesian estimation to determine the robot's pose and a map of the environment given sensor readings, control inputs and the robot's initial pose. Formally, it aims to compute the joint posterior of the robot's pose $\mathbf{x}_k$ and the map $\mathbf{m}$ [50]

$$p(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \propto \mathbf{L} \times p(\mathbf{m}) \tag{4.1}$$

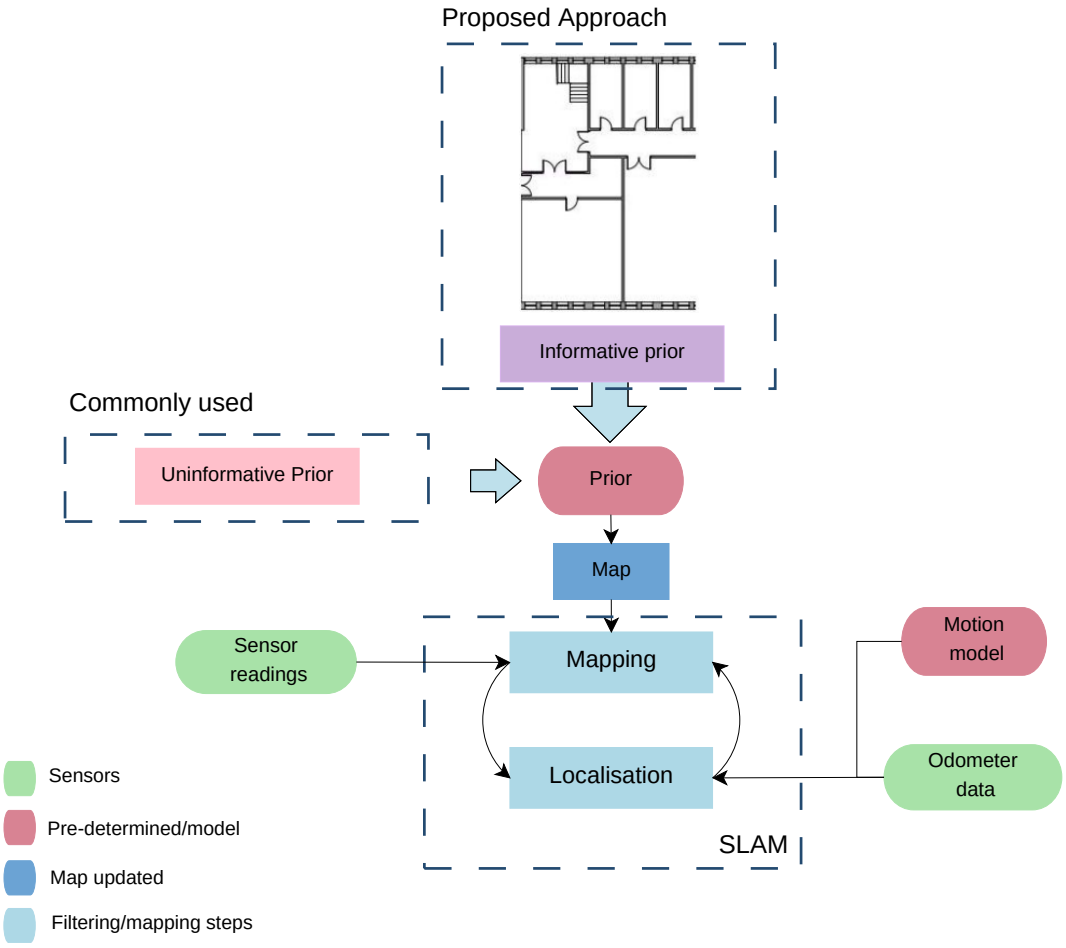


Figure 4.1: The structure of the SLAM problem: sensor data, prior information and motion model predictions are input to the SLAM algorithm that recursively maps the environment and localises the robot within it; architectural drawings and floor plans can be used to extract prior information.

for all times k, where $\mathbf{x}_k$ is the state vector describing the robot's pose, $\mathbf{m}$ is the map of the environment (or a selection of landmarks for feature-based SLAM), $p(\mathbf{m})$ is the prior map, $\mathbf{Z}_{0:k}$ are the sensor observations, $\mathbf{U}_{0:k}$ the history of control inputs, $\mathbf{x}_0$ is the robot pose at time $k = 0$. $\mathbf{L}$ is given by

$$\mathbf{L} = \frac{p(\mathbf{z}_k, \mathbf{x}_k|\mathbf{m})}{p(\mathbf{x}_k, \mathbf{m})} \frac{p(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{p(\mathbf{z}_k|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \tag{4.2}$$

since the joint posterior in (4.1) can be written

$$\begin{aligned} & p(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \\ & p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m}) \frac{p(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{p(\mathbf{z}_k|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \\ = & \frac{p(\mathbf{z}_k, \mathbf{x}_k|\mathbf{m})}{p(\mathbf{x}_k, \mathbf{m})} p(\mathbf{m}) \frac{p(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{p(\mathbf{z}_k|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \\ & \propto \mathbf{L} \times p(\mathbf{m}) \end{aligned} \tag{4.3}$$

indicating that the joint posterior is proportional to the prior probability $p(\mathbf{m})$. Therefore a choice of an optimal prior $p(\mathbf{m})$ can lead to a more accurate estimate of the joint posterior.

The aim is to estimate the robot pose for all times k and the map of the environment given sensor readings, control inputs and a starting pose. SLAM is solved recursively, producing map and pose estimates at each time step. Typically an uninformative prior is used, setting $p(\mathbf{m}_i) = 0.5$, $i = 1, ..., N$, where N is the number of grid cells and $\mathbf{m}_i$ is the ith cell, to indicate that all cells have an equal probability of being occupied or unoccupied. Given the recursive nature of SLAM, using an informative prior map can help reach more accurate solutions after fewer iterations.

In the probabilistic formulation of SLAM the tasks of localisation and mapping cannot be viewed separately. FastSLAM uses Rao-Blackwellization to separate localisation and mapping as discussed in the following section. Therefore FastSLAM allows the study of the effects of a prior map without the need to address localisation.

4.1.2 FastSLAM formulation

Solutions such as FastSLAM [109], DP SLAM [52, 53] and gmapping [66, 67] use particle filters to produce recursive estimates of robot pose and a map of the environment. FastSLAM uses Rao-Blackwellization to decompose the SLAM problem into a robot localisation problem and a collection of landmark/map estimation problems that are conditioned on the robot trajectory estimate [50, 109]

$$\begin{aligned} & p(\mathbf{X}_{0:k}, \mathbf{m} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \\ & = p(\mathbf{m} | \mathbf{X}_{0:k}, \mathbf{Z}_{0:k}) p(\mathbf{X}_{0:k} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) \end{aligned} \tag{4.4}$$

This separation of localisation and mapping allows the study of mapping priors without the need to explore localisation and is thus the implementation used in this thesis.

The aim is to compute the joint posterior of the map and the complete robot trajectory $\mathbf{X}_{0:k}$ rather than the single pose $\mathbf{x}_k$. This is due to the fact that landmarks conditioned on the trajectory are independent, allowing the factorisation shown in Equation 4.4. The particle filter can be used to sample from the motion model and produce the proposal distribution but now the estimates for the landmark locations/the map (conditioned on the robot trajectory estimate) are performed separately.

If a landmark representation is used the location of each landmark can be estimated using a Kalman filter per feature per particle [109]. If an occupancy grid representation is used an occupancy grid map is updated for each particle.

Since the pose and map estimates can be performed separately using Rao-Blackwellization, in occupancy grid SLAM a Monte Carlo Localisation (MCL) algorithm [139] can be implemented to produce pose estimates and calculate $p(\mathbf{X}_{0:k} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$. Then the occupancy grid mapping algorithm can be used to calculate $p(\mathbf{m} | \mathbf{X}_{0:k}, \mathbf{Z}_{0:k})$. The starting pose $\mathbf{x}_0$ is considered to be known for the purposes of this thesis and so prior information about the environment can only be incorporated into $p(\mathbf{m} | \mathbf{X}_{0:k}, \mathbf{Z}_{0:k})$ through the Bayesian prior $p(\mathbf{m})$.

Occupancy grid FastSLAM which uses the occupancy grid mapping algorithm to compute $p(\mathbf{m} | \mathbf{X}_{0:k}, \mathbf{Z}_{0:k})$ is presented in the next section.

4.1.3 Occupancy grid FastSLAM

The occupancy grid FastSLAM algorithm is used in this thesis, implementing the occupancy grid mapping algorithm [51] to update the probabilities of occupancy of each grid cell in the environment. This representation is chosen since the prior map $p(\mathbf{m})$ is the map used at time $k = 0$ which is then recursively updated as occupancy measurements are taken for map cells as the robot explores the environment. In occupancy grid FastSLAM [109] the process of updating pose estimates and updating the map can be performed separately as shown in Equation 4.4. This SLAM format allows for an analysis of the mapping algorithm where prior information is incorporated through the use of a matrix of prior probabilities, $p(\mathbf{m})$.

The occupancy grid mapping algorithm [51] splits the environment to be mapped

into a grid of cells and a prior probability is assigned to each cell. Cells are assumed to be independent in order to update the probability of occupancy of each grid cell. This assumption allows the factorisation

$$p(\mathbf{m}|\mathbf{X}_{0:k}, \mathbf{Z}_{0:k}) = \prod_i p(\mathbf{m}_i|\mathbf{X}_{0:k}, \mathbf{Z}_{0:k}) \tag{4.5}$$

where $i = 1, ..., N$ is the current grid cell and N the total number of grid cells. Updating the probability for each cell given the robot pose and sensor readings is then merely an update of the probability of occupancy of that cell. As the robot moves through the environment the probabilities of occupancy of cells that are within the field of view of the robot sensors are updated.

The probability that a cell $\mathbf{m}_i$ is occupied given the observation history is given by

$$p(\mathbf{m}_i|\mathbf{Z}_{0:k}) = \frac{p(\mathbf{z}_k|\mathbf{m}_i, \mathbf{Z}_{0:k-1})p(\mathbf{m}_i|\mathbf{Z}_{0:k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{0:k-1})} \tag{4.6}$$

which can be written as

$$p(\mathbf{m}_i|\mathbf{Z}_{0:k}) = p(\mathbf{z}_k|\mathbf{m}_i)\frac{p(\mathbf{m}_i|\mathbf{Z}_{0:k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{0:k-1})} \tag{4.7}$$

using the static world assumption which states that past sensor readings are conditionally independent given knowledge of the map $\mathbf{m}$ [136]. Thus Equation 4.6 can be written as

$$p(\mathbf{m}_i|\mathbf{Z}_{0:k}) = \frac{p(\mathbf{m}_i|\mathbf{z}_k)p(\mathbf{z}_k)}{p(\mathbf{m}_i)}\frac{p(\mathbf{m}_i|\mathbf{Z}_{0:k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{0:k-1})} \tag{4.8}$$

using $p(\mathbf{z}_k|\mathbf{m}_i) = \dfrac{p(\mathbf{m}_i|\mathbf{z}_k)p(\mathbf{z}_k)}{p(\mathbf{m}_i)}$.

Placing Equation 4.8 in the odds form we get

$$\begin{aligned} odds(\mathbf{m}_i|\mathbf{Z}_{0:k}) &= \frac{p(\mathbf{m}_i|\mathbf{Z}_{0:k})}{p(\neg\mathbf{m}_i|\mathbf{Z}_{0:k})} \\ &= \frac{p(\mathbf{m}_i|\mathbf{z}_k)p(\mathbf{m}_i|\mathbf{Z}_{1:k-1})p(\neg\mathbf{m}_i)}{p(\neg\mathbf{m}_i|\mathbf{z}_k)p(\neg\mathbf{m}_i|\mathbf{Z}_{1:k-1})p(\mathbf{m}_i)} \end{aligned} \tag{4.9}$$

Therefore

$$\begin{aligned} odds(\mathbf{m}_i|\mathbf{Z}_{0:k}) = \\ odds(\mathbf{m}_i|\mathbf{z}_k)odds(\mathbf{m}_i|\mathbf{Z}_{1:k-1})(odds(\mathbf{m}_i))^{-1} \end{aligned} \tag{4.10}$$

Where $p(\mathbf{m}_i)$ is the prior probability of occupancy of the ith grid cell,

$odds(\mathbf{m}_i|\mathbf{Z}_{1:k-1})$ is the odds at the previous time step and $odds(\mathbf{m}_i|\mathbf{z}_k)$ represents the inverse sensor model. Taking the logarithm of Equation 4.9, the log odds representation of Equation 4.9 is

$$l_{k,i} = InverseSensorModel(\mathbf{m}_i, \mathbf{x}_k, \mathbf{z}_k) + l_{k-1,i} - l_{0,i} \tag{4.11}$$

with

$$l_{k,i} = log\frac{p(\mathbf{m}_i|\mathbf{Z}_{1:k}, \mathbf{x}_{1:k})}{1 - p(\mathbf{m}_i|\mathbf{Z}_{1:k}, \mathbf{x}_{1:k})} \tag{4.12}$$

Research has been conducted to identify suitable or preferable sensor models such as [136] and [157].

The Bayesian priors for each cell, $p(\mathbf{m}_i)$ are incorporated through $l_{0,i}$, the log odds prior for a given cell. When there is no available prior information a non-informative Bayesian prior is assigned to all grid cells, $p(\mathbf{m}_i) = 0.5,\ i = 1, ..., N$. Most researchers use such a prior in order to produce solutions that do not depend on having prior knowledge of the environment [50]. Others claim that access to information like detailed architectural drawings may be difficult [93]. Information such as floor plans for buildings like hospitals and offices is generally available, however, and can be used to extract useful information and construct SLAM priors as described in Chapter 3.

Incorporating prior information does not add to the computational cost of running SLAM and only incurs a one-off cost of extracting prior information and constructing a prior. Since the posterior is proportional to the prior map (Equation 4.3) and given the recursive nature of SLAM, constructing an informative prior map $p(\mathbf{m})$ can help produce a more accurate map even if a quick exploration is performed. In the case of a quick exploration each grid cell may only be scanned once or twice, making the effect of the prior more significant. This is especially useful for time critical applications where the environment needs to be explored quickly such as USAR missions.

4.2 Problem formulation

Using the setup described in the previous section this chapter proposes a method to shortlist optimised prior maps, $p(\mathbf{m})$, leveraging the information extracted from an architectural drawing or floor plan with a method such as the one proposed in Chapter 3.

Once wall locations have been extracted from an architectural drawing, prior occupancy values $p(\mathbf{m}_i)$ need to be assigned to each grid cell $\mathbf{m}_i$ accordingly. Three parameters have to be chosen:

- Prior probability assigned to detected walls
- Prior probability assigned to detected empty space
- Occupancy grid cell size (grid resolution)

Each grid cell $\mathbf{m}_i$ is assigned a Bayesian prior probability $p(\mathbf{m}_i)$ as discussed in Section 4.1.3. The following notation will be used throughout the rest of this thesis. A high probability assigned to a grid cell will mean the cell has a high probability of being empty. Conversely, a low probability will signify a low probability that a cell is empty. If wall locations are known, cells that are located where walls were detected can be assigned lower prior probabilities, $p(\mathbf{m}_i^{wall}) < 0.5$, and cells located where empty space was detected can be assigned higher prior probabilities, $p(\mathbf{m}_i^{space}) > 0.5$. For brevity these two probabilities will be written p_{wall} and p_{space}, where p_{wall} is the prior probability assigned to grid cells that correspond to locations of detected walls in the architectural drawing and p_{space} the prior probability assigned to cells corresponding to detected empty space in the drawing. This notation can be used since all grid cells $\mathbf{m}_i$ that correspond to occupied locations will be assigned a prior probability of p_{wall} and those that correspond to empty space a prior probability p_{space}.

The alignment of grid cells and detected walls depends on the occupancy grid cell size. In all the results presented an appropriate grid cell size that yields a reasonable compromise between accuracy and computational complexity was chosen for all drawings used. The effects of using priors when coarser grids are used is discussed in the results section.

The problem addressed in this chapter is therefore a study of different possible combinations of (p_{wall}, p_{space}) to determine the pair or pairs of values that optimise the chosen performance metrics described in the next section. Therefore, the design parameters to determine are

$$0 \leq p_{wall} \leq 1 \;\; and \;\; 0 \leq p_{space} \leq 1 \tag{4.13}$$

In order to assess the effects of using different prior parameter pairs (p_{wall}, p_{space}) a measure of quantitative performance is proposed in the next section.

4.3 Quantitative performance assessment

A number of different approaches have been proposed to assess map quality but there is no established method to do so [36, 65, 84, 103, 105]. A novel approach

to assess the quality of maps produced using a mapping algorithm is proposed in this section, using a classification problem formulation.

4.3.1 Robot map evaluation background

The lack of a widely accepted and recognised method to assess performance makes it difficult to compare the results of different methods used by different researchers and research groups. Some attempts have been made, however, to quantitatively assess maps, presented below.

Consistency Map consistency is a measure of map performance but, according to [105], there is no consistent notion of consistency and determining whether or not a SLAM map is consistent is still an open problem. Global consistency is often used to mean that the map produced agrees with the ground truth whereas local consistency refers to correctly aligning sensor scans locally. A measure for global consistency is proposed in [105] which uses the mismatch in the sensor data but this framework is tailored to SLAM using 2D laser sensors and can struggle in dynamic environments.

Accuracy Assessing map accuracy using a quantitative measure of performance has been proposed in [84]. The metric used is dependent on the difference between the ground truth and the produced map at a number of defined control points in the map and the localisation error at those points.

Occupancy grid maps A number of methods have been proposed to assess the quality of occupancy grid maps. Some methods such as [36] compute a correlation between the produced map and ground truth. A problem with this approach is that a misalignment between the true environment and the map can lead to a very low score but also calculating correlations can be computationally expensive. Another method, Map Score [103], is based on the probability that two maps represent the same world and checks the agreement between the two. However, this type of approach tends to be biased towards free cells since maps contain much more free space than occupied areas. Finally, as discussed in [65], the percentage of free and occupied cells identified correctly can be used to measure performance.

Maps in simulated environments Some papers have proposed metrics to evaluate the performance of teams taking part in robot competitions. A set of metrics to assess performance of robots competing in RoboCup search and rescue is proposed in [22]. The authors claim these metrics could be used to evaluate USAR

robots operating in a real environment, however the aims and objectives of the competition may not necessarily be priorities in a real life operation. For example, the time the system takes to run and power consumed are factors that are not taken into account in the performance metric.

These methods are either tailored to a specific application and hence cannot be easily used in this case, too computationally expensive or depend on localisation accuracy. Given these limitations a new approach is proposed in the next section that addresses these problems.

4.3.2 Proposed approach

In this chapter a metric that is easy to evaluate is proposed, that gives an indication of how successful the system is in detecting occupied and empty space in the environment. This leads naturally to a classification problem formulation, classifying each pixel in the image as belonging to one of two categories: occupied or empty space.

The use of the percentage of free and occupied cells identified correctly is proposed as a performance metric in [65]. The novelty of the proposed approach is the formulation of the assessment of the quality of an occupancy grid map as a classification problem where the aim is to correctly classify pixels as corresponding to occupied or unoccupied space. Therefore a precision-recall analysis can be used to evaluate the quality of the produced maps. Occupied locations in the environment can then be defined as true positives and empty space as true negatives. Any other objects detected can then be defined as false positives.

The chosen metrics are therefore detailed as follows [39]

$$\text{Precision } (pre) = \frac{TruePositives}{DetectedOccupiedSpace} = \frac{TP}{TP + FP}$$

$$\text{Recall } (rec) = \frac{TruePositives}{ActualOccupiedSpace} = \frac{TP}{TP + FN}$$

with
TP = number of true positives = walls
FP = number of false positives = empty space mapped as walls
FN = number of false negatives = walls mapped as empty space

Precision represents the percentage of correctly detected walls and recall represents the number of walls correctly detected out of all walls in the drawing. These

metrics were chosen because between them they represent the two most important aspects of the final map. A high precision is necessary to ensure the detected occupied space actually corresponds to walls. A high recall is also required to ensure that the system has managed to correctly map the majority of walls in the environment.

The false positive rate (fpr) metric used in Chapter 3 is not used in this case because a high precision was found to yield a low false positive rate. The false positive rate is defined as

$$\text{False positive rate } (fpr) = \frac{FalsePositives}{ActualNegatives} = \frac{FP}{TN + FP}$$

where TN is the number of true negatives, true empty space in the environment. In most environments $TN >> FP$ even for a poor mapping system, since most buildings are made up mostly of empty space/non-walls, resulting in low fpr values. False positive rate values observed during testing were typically $< 20\%$ even for very low grid resolutions. Moreover, a higher precision was found to result in a lower fpr and vice versa. This is not surprising given

$$fpr = \frac{FP}{TN + FP} = \frac{1}{\frac{TN}{FP} + 1} \text{ and } pre = \frac{TP}{TP + FP}$$

with a low FP resulting in a higher precision and lower false positive rate.

Following these assessment criteria, each point in the robot map is tested against the true environment to determine whether it has been classified correctly. A simple way to perform this comparison in practice is to find the difference between the true environment image and the produced map to identify false positives and false negatives and thus calculate the above metrics. This difference between the ground truth, I, and the final map, $\mathbf{m}$, is defined as

$$d = I - \mathbf{m} \tag{4.14}$$

Then TP and TN are determined from I and FP is calculated using

$$FP = \{d,\ for\ d \geq 0.8\} \tag{4.15}$$

This signifies a wall was detected where there is an empty space since for $d_i \geq 0.8$ we have

$$d_i = I_i - \mathbf{m}_i \geq 0.8 \Rightarrow I_i \geq \mathbf{m}_i + 0.8 \tag{4.16}$$

for the ith location.

Moreover, since $\mathbf{m}_i \geq 0$

$$0 \leq I_i \leq 1 \Rightarrow 0 < \mathbf{m}_i + 0.8 \leq 1 \Leftrightarrow -0.8 < \mathbf{m}_i \leq 0.2 \tag{4.17}$$

but since $0 \leq \mathbf{m}_i \leq 1$, Equation 4.17 becomes

$$0 \leq \mathbf{m}_i \leq 0.2 \tag{4.18}$$

signifying this location was mapped as a wall. The value 0.8 was empirically determined to yield realistic results: increasing this value would require very confident empty space detection which is an unrealistic requirement and decreasing it was found to reduce the separation between detected walls values and detected empty space values, making the assessment more prone to errors. This is a simple but effective and computationally inexpensive method to assess quantitative performance.

As shown in the results section, constructed priors were tested in a simulation which assumes perfect knowledge of the robot's location so over or underestimating these values if the map and ground truth images are misaligned is not a problem. Therefore this method of comparing the map to ground truth to find the number of true/false positives/negatives is used to test the performance of maps produced using different priors.

If tests are conducted on a point-by-point basis coarse grids can be unjustly penalised. This would result in an area of misclassified points that are actually not strictly a classification error, just a limitation of an unsuitable grid resolution. In order to avoid penalising a system with an unsuitably coarse grid resolution, maps that perform well quantitatively are also tested qualitatively to ensure overall good performance.

While testing different prior maps for different drawings it was observed that the objectives of maximising precision and maximinsing recall are conflicting. Therefore selecting optimal prior values to maximise both objectives is not trivial and a multi-objective optimisation is proposed to determine optimised prior values.

4.4 Contextual prior optimisation

Using the method described in Chapter 3 walls and empty space can be detected. Prior values of occupancy then need to be assigned to each grid square. In order to determine which (p_{wall}, p_{space}) pair yields optimal performance a multi-objective optimisation can then be performed to determine values that yield both maximum

precision and maximum recall.

In order to test how the values for (p_{wall}, p_{space}) that yield maximum precision and recall compare in a qualitative sense, maps can be constructed using the prior (p_{wall}, p_{space}) values yielding maximum precision and recall. These can then be tested to assess the qualitative performance of maps produced using these prior values. If requirements for each metric are conflicting a multi-objective optimisation can be performed to determine pairs $\boldsymbol{\pi} = (p_{wall}, p_{space})$ that yield both high precision and high recall. Having both precision and recall $> 40\%$ was empirically determined to be an acceptable threshold. Given the multi-objective nature of the problem, having both precision and recall higher than 50% may not always be achievable; 40% was found to be the highest percentage that could be achieved by both recall and precision. This threshold was chosen to avoid discarding all solutions in case of an incomplete exploration or a coarser grid representation.

This multi-objective problem can be formulated as

$$\begin{aligned} \min_{\boldsymbol{\pi}} F(\boldsymbol{\pi}) &= [f_{pre}(\boldsymbol{\pi}) \;\; f_{rec}(\boldsymbol{\pi})] \\ subject\ to\ C &= \begin{cases} 0.1 \leq \boldsymbol{\pi} \leq 1 \\ f_{pre}(\boldsymbol{\pi}) < 2.5 \\ f_{rec}(\boldsymbol{\pi}) < 2.5 \end{cases} \end{aligned} \tag{4.19}$$

where $\boldsymbol{\pi} = (p_{wall}, p_{space})$, $f_{pre}(\boldsymbol{\pi}) = \frac{1}{pre}$ and $f_{rec}(\boldsymbol{\pi}) = \frac{1}{rec}$ (so $f_{pre}(\boldsymbol{\pi}), f_{rec}(\boldsymbol{\pi}) < 2.5 \Leftrightarrow pre, rec > 40\%$). The value range $[0.1, 1]$ was chosen for the probabilities $\boldsymbol{\pi} = (p_{wall}, p_{space})$ to avoid probability values too close to 0 that would lead to numerical instabilities.

In order to perform this multi-objective optimisation a Pareto-based method was chosen since the relative importance of the objectives is unclear [62]. In this minimisation problem a decision vector $\hat{\boldsymbol{\pi}}$ with $\hat{\boldsymbol{\pi}} \in C$ is Pareto optimal if there is no other $\boldsymbol{\pi} \in C$ for which $f_i(\boldsymbol{\pi}) \leq f_i(\hat{\boldsymbol{\pi}})$, $\forall i$ and at least one $f_i(\boldsymbol{\pi}) < f_i(\hat{\boldsymbol{\pi}})$ for $i = 1, ..., k$ where k is the number of functions in $F(\boldsymbol{\pi})$. In this case the decision vector $\hat{\boldsymbol{\pi}}$ is said to Pareto-dominate vector $\boldsymbol{\pi}$. If only the second condition is met the solution is considered weakly Pareto optimal. The multi-objective optimisation solver used aims to find a subset of Pareto optimal solutions which is referred to as the Pareto front [62], Figure 4.2.

The concept of Pareto dominance is applied in order to use a genetic algorithm to solve this multi-objective problem. First the objective function is evaluated for each individual in the population $\boldsymbol{\Pi}$. Non-dominated individuals, $\boldsymbol{\Pi}_{nond}$ are then found and removed from the population. This process is repeated until all non-

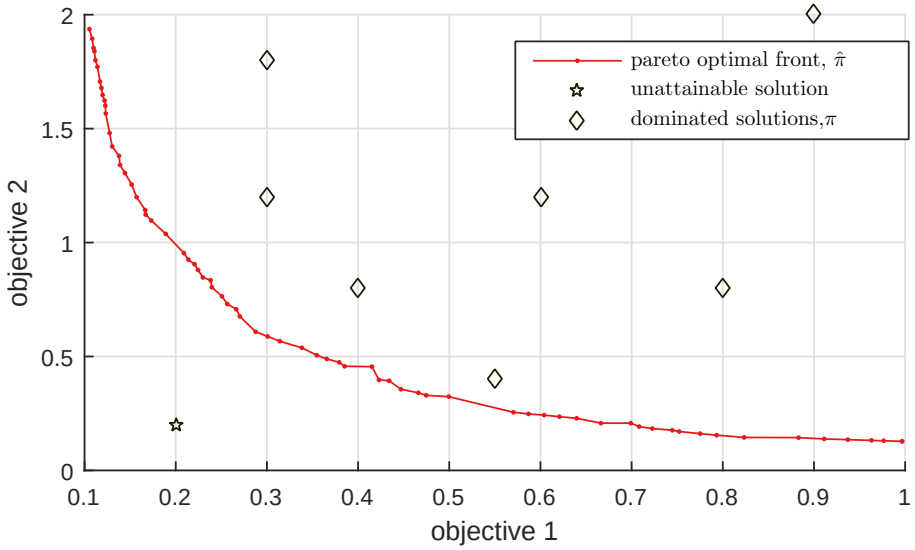


Figure 4.2: An example of a Pareto optimal front, for the problem of minimising objectives 1 and 2. Pareto optimal values $\hat{\boldsymbol{\pi}}$ dominate the rest of the $\boldsymbol{\pi}$ values. Solutions to the right of the Pareto front are dominated and solutions to the left of the front are unattainable.

dominated individuals $\mathbf{\Pi}_{nond}$ have been identified. A controlled elitist genetic algorithm [43] (a variant of NSGA-II [44]) was chosen since it is computationally cheaper than other genetic algorithms and is able to identify a set of solutions that has improved spread and better convergence near the true Pareto front [44].

For a solution p the number of solutions that dominate it, n_p, and the number of solutions that are dominated by it, d_p, are calculated. Solutions with $n_p = 0$ are non-dominated for this set, and values with low n_p among their neighbours are also shortlisted [44]. A random parent population P_0 is used to initialise the genetic algorithm and the population is sorted based on non-domination. Solutions are thus assigned a rank based on their non-domination value, with lower values signifying better performance and a rank of 1 being optimal. The aim is then to minimise rank to shortlist non-dominated solutions.

In order to maintain diversity in the population of shortlisted values a distance measure between the current solution and its neighbours is used, as described in [44]. This measure assesses how crowded a solution is based on how many close neighbours it has and allows the removal of solutions that fall in very crowded areas. This method thus avoids having a large number of solutions all clustered close together.

In order to obtain recall and precision variables and perform the multi-objective optimisation a simulation environment was devised as detailed in the next section.

4.5 Simulation setup

A simulation was used to produce maps using different (p_{wall}, p_{space}) pairs. A simulation allows the testing of a large number of possible prior values to assign to (p_{wall}, p_{space}). Thus the multi-objective optimisation can be performed without the need to obtain mapping results for different buildings and different building drawings using a robot which would be very time consuming. The drawing test set, simulation used and assumptions made are discussed in this section.

4.5.1 SLAM code and datasets background

Simulated vs real data

A number of SLAM data sets are available and commonly used to test different SLAM algorithms [2, 143]. These datasets provide raw sensor and odometry data collected by exploring buildings or outdoor areas that can be used to test different SLAM algorithms. Distance sensor data and odometry values for indoors environments are required in this thesis, limiting the number of appropriate available datasets. Moreover, the number of available indoors datasets using distance sensors is small, limiting the variety of possible architectural drawings and floor plans that can be used as a source of prior information and requiring the appropriate plans to be obtained. Conversely, generating appropriate data sets for a sufficient and diverse number of building and drawing combinations would be challenging and very time-consuming and is thus outside the scope of this thesis. In order to be able to explore a number of diverse and representative drawings from which to extract prior information a simulated environment is chosen to explore the benefits of using informative Bayesian priors for indoors environments.

Existing SLAM software

The most commonly used ROS [3] module that uses an occupancy map representation is an implementation of gmapping that uses a tree structure to update values of occupancy and assumes each square can be empty, occupied or unknown [61]. Instead of using a recursive probabilistic method to update continuous values of occupancy, binary occupancy values are updated based on sensor readings. Therefore this type of representation does not lend itself to a study of the effects of using informative priors.

OpenSLAM [2] also provides some occupancy grid SLAM implementations including a version of gmapping [66], an implementation of DP-SLAM [53] and an implementation of GridSLAM [69]. The DP-SLAM implementation presents

the same limitations as the gmapping implementation, using a tree structure to store binary values of occupancy for explored squares only. Finally, the GridSLAM code uses outdated libraries and would require much restructuring to produce a working version, making producing a full working version of this code a project diverging from the main aims of the thesis.

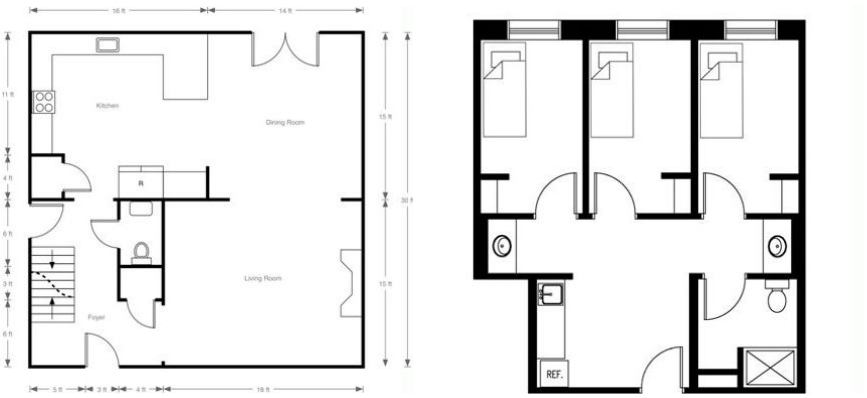
Given the shortcomings of available SLAM code that is suitable to test the proposed approach an occupancy grid mapping simulator for Matlab was produced. This simulator does not address the localisation aspect of SLAM but assumes known robot poses. Given the separation of the localisation and mapping tasks in FastSLAM, however, as discussed in Section 4.1.2 this was considered sufficient to test the effects of different prior values assigned to detected walls and empty space in drawings. This approach allows for the study of a large number of different prior value $(p_{wall},\ p_{space})$ pairs for a number of different drawings and simulated environments. Using Matlab also facilitates the analysis of obtained results using standardised and easily available libraries. Finally, the simulator produced can also be used as an educational tool to help visualise the effects of recursive probabilistic mapping and the effects of using different prior maps.

4.5.2 Proposed approach

Given the limitations of existing SLAM code, a simulator was produced to study the effects of using informative priors. Simulated sensor data were used to allow a study of the effects of using priors extracted from different drawings, yielding a diverse set of case studies.

Drawing test set selection

A number of different drawings were tested using the simulation environment described later in this section. A version of each drawing containing only structural information was created by hand and used as the ground truth. Representative drawings and relevant results are presented in this chapter. Two floor plans, drawings (a) and (b) Figure 4.3, are presented. These were chosen as a representative sample of common elements and configurations found in architectural drawings and floor plans. Drawings (a), (b) and (c) in Figure 4.4, are sections of an architectural drawing of a University of Sheffield Engineering building. These sections were chosen because they are a representative sample of different outlines and commonly observed elements. Drawing (b) contains only doors and wall segments making it an easy drawing to process and drawings (a) and (c) are quite challenging to process, containing stairs, text, walls of varying width and, in the



(a) (b)

Figure 4.3: Image test set floor plans. Drawing (a) contains labelling, thin and thick lines, stairs, double doors and inbuilt furniture, drawing (b) contains suggested furniture and details.

case of drawing (c), a door at an angle.

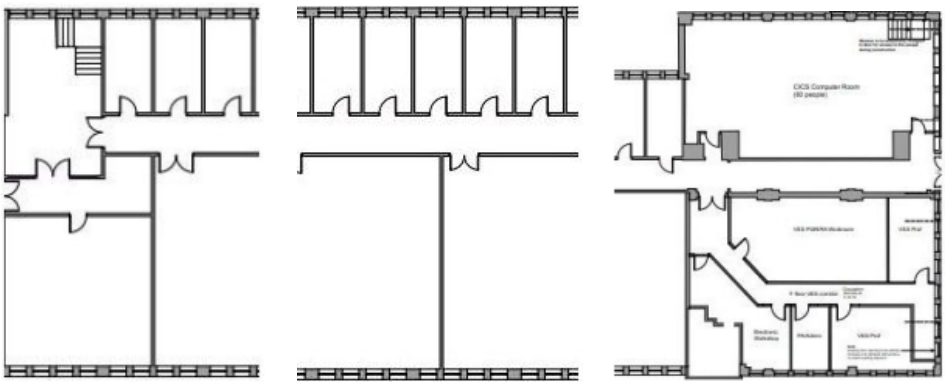
Occupancy grid mapping simulation

In order to test the effects of using informative priors on map quality, a simulation environment was created in Matlab. In this simulation the robot pose is assumed to be known and accurate. Given the separation of localisation and mapping using Rao-Blackwellization performed in FastSLAM (Section 4.1.3), mapping can be studied separately by providing deterministic pose values instead of using Monte Carlo localisation to produce pose estimates. Mathematically, instead of using Equation 4.4 and producing estimates of both the map and trajectory for each particle we assume $p(\mathbf{X}_{0:k}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$ is known and hence only one map needs to be updated by determining $p(\mathbf{m}|\mathbf{X}_{0:k}, \mathbf{Z}_{0:k})$ using the occupancy grid mapping algorithm [51].

A robot is modeled as a point moving through space and the robot posse $\mathbf{x}_k$ at each time step k is assumed to be known and accurate, given by

$$\mathbf{x}_k = (x, y, \theta) \tag{4.20}$$

where (x, y) give the robot position in Cartesian coordinates and θ the robot orientation. Four possible θ values are modeled, $\theta = [0, \frac{\pi}{2}, \pi, \frac{3\pi}{4}]$. An $s \times 3$ array is passed into the simulator, where s is the number of (x, y, θ) readings. No motion model is used to construct these since they are assumed to be known- and perfect. Enough s readings to obtain distance sensor readings that cover the



(a) (b) (c)

Figure 4.4: Image test set architectural drawing sections of one of the University of Sheffield's Engineering buildings. (a) contains stairs and double doors, (b) is a more straightforward drawing to process and (c) presents a number of difficulties including text, stairs, labelling and a door at an angle.

building explored in each case were used (not all collections of s readings used to update the map correspond to complete trajectories). This is effectively down-sampling the amount of motion and distance data points used to update the map. The simulated distance sensor is a Microsoft Kinect sensor with a conical field of view defined by an angle ϕ and range r.

A value of $\phi = \pi/6$ was used in all simulations. The drawings used are not all drawn to the same scale and so the value of r in pixels was set by using the size of doors as a way to determine scale and ensure consistency between drawings. The value of r was set to be the same as the length of the door side in each drawing, setting it to 20 pixels for the architectural drawings and 40 pixels for the floor plans.

At each time step, cells that fall within the field of view of the sensor are determined. This is achieved by calculating the equations of two lines and arc that make up the sensor cone for the robot's current pose and finding cells that fall within the cone defined by these three curves.

The occupancy grid mapping algorithm [51] is used to update the log odds of occupancy $l_{k,i}$ for each grid cell within the field of view of the sensor at each time step (Section 4.1.3). The probabilities of occupancy can be retrieved from $l_{k,i}$ using

$$p(\mathbf{m}_i|\mathbf{Z}_{1:k}, \mathbf{x}_{1:k}) = 1 - \frac{1}{1 + \exp(l_{k,i})} \tag{4.21}$$

The following inverse sensor model was used, based on [127]. For each grid cell $\mathbf{m}_i$, within the field of view

$$InverseSensorModel(\mathbf{m}_i, \mathbf{x}_k, \mathbf{z}_k) = \begin{cases} 0.9, & n_i^{pixels<z} = n_i^{pixels} \\ 0.1, & n_i^{pixels>z} = n_i^{pixels} \\ \dfrac{n_i^{pixels<z}}{n_i^{pixels}}, & 0 < n_i^{pixels<z} < n_i^{pixels} \end{cases} \tag{4.22}$$

with $n_i^{pixels<z}$ being the number of map pixels for a cell i that fall between the robot and the location of the detected obstacle, z; $n_i^{pixels>z}$ the number of map pixels for a cell i that are located beyond the detected object or at the location of the detected object; and n_i^{pixels} the number of map pixels in cell i. For an accurate sensor, cells for which all map pixels correspond to detected empty space are assigned a value of 0.9, those that correspond to detected objects a value of 0.1 and those that contain both empty space and objects are assigned a value equal to the percentage of pixels corresponding to detected objects.

When prior information is available $p(\mathbf{m}_i)$ can be assigned a different prior value based on whether a wall or empty space was detected for grid cell i during the drawing processing stage as discussed in Chapter 3. The results obtained using this simulator and performing multi-objective optimisation to determine optimal prior values (p_{wall}, p_{space}) are presented in the next section.

The main functionalities of the simulator are as follows:

- The trajectory of the robot as defined by an array of (x, y, θ) values can be input by the user or loaded from a file
- An uninformative or an informative prior can be used
- Grid resolution, sensor range and field of view and prior values assigned to walls and empty space in the prior map can all all be easily adjusted
- The map is displayed as the robot explores the environment, allowing visualisation of results
- Calculation of precision and recall metrics for the final map
- The development of an overall algorithm to construct optimised Bayesian priors using an architectural drawing or floor plan that result in more accurate maps without adding to the computational cost of running the SLAM algorithm

4.6 Results and discussion

This section presents the results obtained using the simulator presented in Section 4.5.2, testing different prior values to assign to detected walls, p_{wall}, and to detected empty space, p_{space}. The results of the multi-objective optimisation performed to determine the (p_{wall}, p_{space}) pair that yields optimised precision and recall are also presented.

In order to evaluate the effect of different priors different combinations of prior values (p_{wall}, p_{space}) were examined. The simplest way of assigning (p_{wall}, p_{space}) is to assign $p_{space} = 1$ and $p_{wall} = 0$. However, there are a number of problems with this approach. Firstly, assigning a 1 or a 0 causes numerical instabilities when log odds are used to update the occupancy grid map. Secondly, starting from very confident initial predictions makes it more difficult for the system to correct a prior value based on sensor data. In order to avoid any numerical instabilities all priors tested were multiplied by a factor of 0.9 and 0 was not included in the set of prior values tested.

Given these limitations the values tested for (p_{wall}, p_{space}) were in the range of $[0.1, 1]$. A set of discrete values were tested within this range, spaced 0.1 apart. This set of discrete values was chosen to reduce the computation required to test all possible combinations of (p_{wall}, p_{space}) within this range. Continuous values of precision and recall were then obtained using linear interpolation, with the spacing of 0.1 between data points being sufficiently small to allow for interpolation to yield reasonable results.

Figure 4.5 shows the precision and recall colour maps for all representative drawings. These figures indicate that the objectives of maximising precision and maximising recall are conflicting for this problem, with regions of high precision (yellow areas) corresponding to regions of low recall (blue areas) and vice versa.

In the precision plots of Figure 4.5B we observe higher values for $p_{wall} = 0.2$. That is because 0.2 is the lowest probability value for which the log odds value is in the linear section of the log odds plot, Figure 2.9. Therefore it is the lowest value we can assign to detected walls that avoids the region close to the asymptote near 0.

Figure 4.6(c) shows colourmaps of the precision and recall metrics for possible combinations of p_{wall} and p_{space} for a representative drawing, Figure 4.6(a). Linear interpolation was used to produce continuous metric values. Figure 4.6(b) shows the final maps produced using the prior values that yield optimal precision and those that yield optimal recall.

A higher precision ensures that it is unlikely free space will be incorrectly

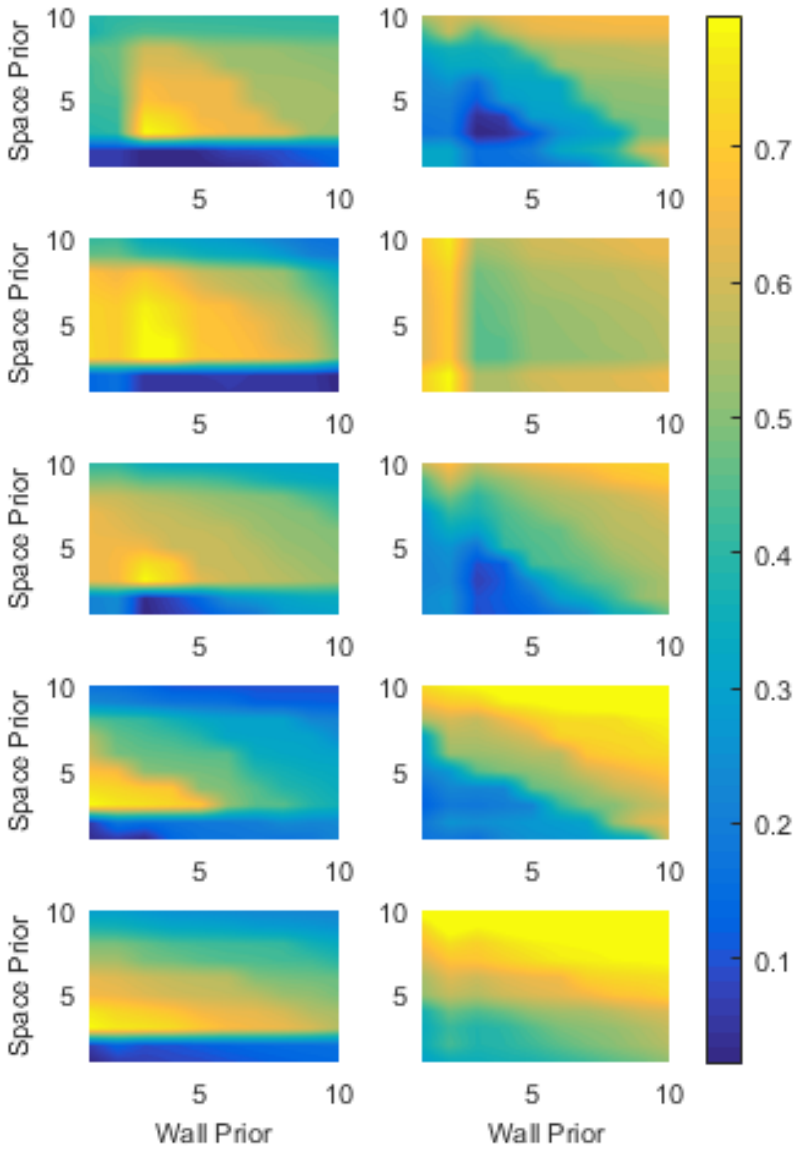
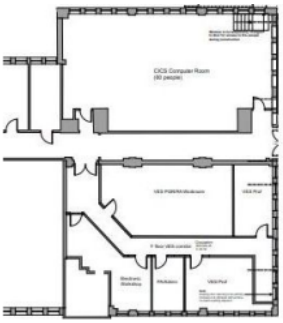
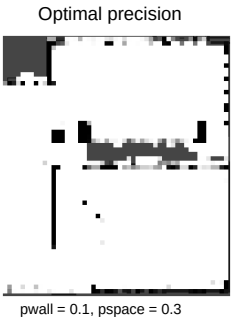
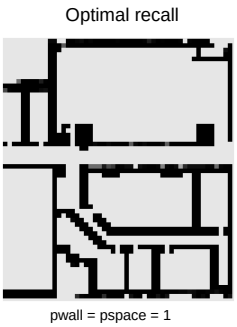


Figure 4.5: Interpolated precision and recall colour maps for the five drawings, with rows 1 and 2 being floor plans and rows 3,4 and 5 being sections of an architectural drawing. Regions of high precision, shown in yellow, correspond to regions of low recall, shown in blue, and vice versa. This indicates that the objectives of maximising precision and maximising recall are conflicting and thus a multi-objective optimisation is required to identify optimal prior parameters.

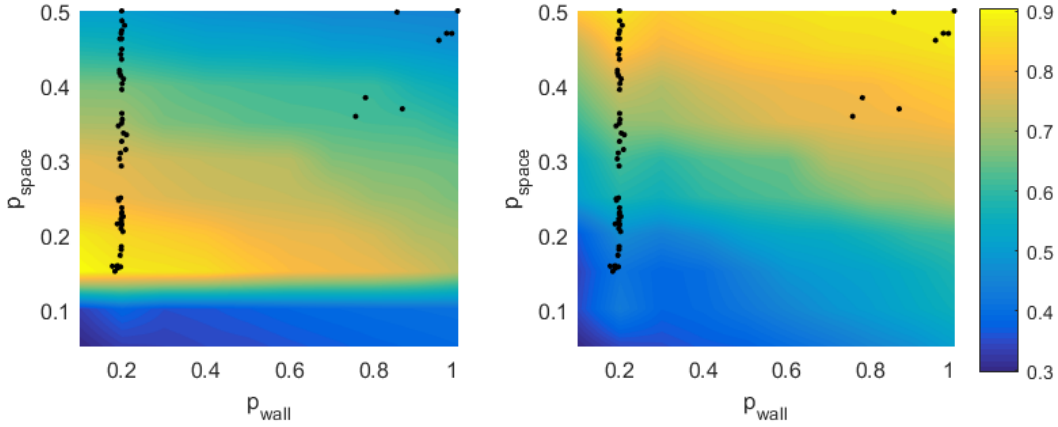


(a) Original architectural drawing

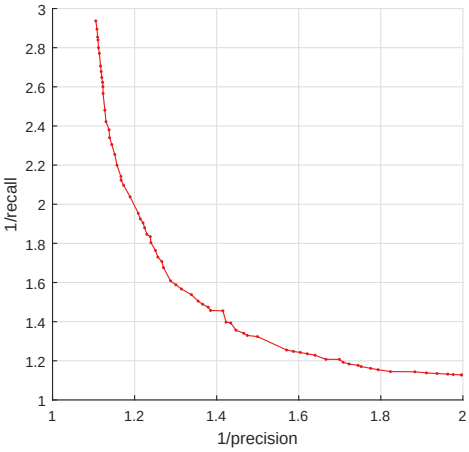




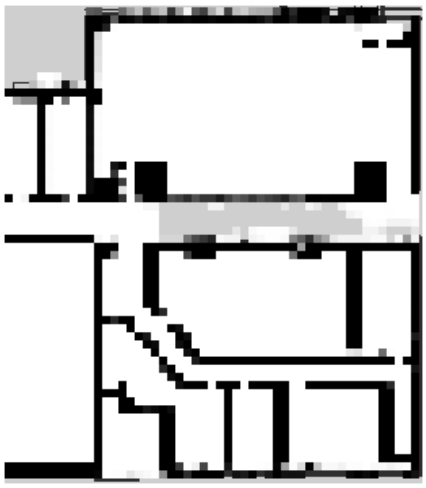
(b) Maps produced using a prior that maximises a single objective



(c) Precision and recall metrics



(d) Multi-objective optimisation: Pareto front



(e) Map produced using prior optimising conflicting objectives

Figure 4.6: Multi-objective optimisation overview: (a) is the architectural drawing used to extract prior information; (b) shows the maps produced using prior values that yield maximum precision and maximum recall, neither of which is qualitatively optimal; (c) shows the *pre* and *rec* colour maps for all combinations of (p_{wall}, p_{space}) between 0.1 and 1 and the Pareto optimal solutions as black points; (d) shows the Pareto front produced by the multi-objective otpimisation and (e) shows the map produced using the Pareto optimal proposed prior values $p_{wall} = 0.2,\ p_{space} = 0.9$.

identified as occupied and a higher recall ensures that the building structure is detected. Figure 4.6(c) highlights the fact that the aims of having high precision and also maintaining a high recall are conflicting. This problem of determining a suitable pair of (p_{wall}, p_{space}) that ensures high precision and high recall ($> 40\%$ was empirically determined to be an appropriate threshold, as was explained in Section 4.4) is therefore a multi-objective optimisation problem.

The Pareto front for a representative drawing, Figure 4.6(a), can be seen in Figure 4.6(c) and the final map produced using one of the Pareto optimal prior values is shown in Figure 4.6(e). This drawing was chosen because, due to the level of detail including labelling, walls at an angle and multiple rooms, it highlights the qualitative difference in performance between Figure 4.6(b) and Figure 4.6(e).

The map shown in Figure 4.6(e) avoids detecting thicker walls or missing doors as can happen using the prior values that optimise recall, Figure 4.6(b), and also avoids missing most of the walls as is done using the prior values that yield maximum precision, Figure 4.6(b).

Figure 4.5 shows the precision and recall colour maps for all representative drawings. The shape of the colour maps for precision and recall does not vary greatly between drawings, other than the fact that the region of values that yield good performance is larger/smaller for different drawings. Small differences are observed, such as the higher recall values observed for drawing (b) for p_{wall} values between 0.1 and 0.3. However, there are enough similarities in these results to indicate that a globally optimal region of (p_{wall}, p_{space}) values that result in high precision can be identified.

The locations of maximum precision for three different grid resolutions are shown in Table 4.1. Different grid resolutions are tested to examine whether prior values that maximise precision and recall vary greatly with grid cell size. Maximum values are consistently observed for p_{wall} values between 0.1 and 0.3 and $p_{space} = 0.3$ regardless of the drawing and grid resolution with the exception of drawing (a) for a 10x10 grid cell.

The values that yield optimal recall can be seen in Table 4.2. These values are in most cases what one would intuitively expect to be optimal: a high value for p_{space} and a lower value for p_{wall}. For the architectural drawing sections, optimised values were found to be $p_{space} = 1$ with p_{wall} between 0.1 and 1.

As shown in Tables 4.1 and 4.2 the values that yield maximum precision do not correspond to values that yield maximum recall and vice versa, confirming the fact that the problem of maximising both precision and recall is a multi-objective optimisation problem.

Multi-objective optimisation using the controlled elitist genetic algorithm [43]

Grid cell 3x3		
Drawing	Max pre %	(p_{wall}, p_{space})
a	98.49	(0.3, 0.3)
b	99.42	(0.3, 0.3)
c	91.77	(0.3, 0.3)
d	95.56	(0.1, 0.3)
e	93.77	(0.3, 0.3)
Grid cell 5x5		
a	75.00	(0.3, 0.3)
b	95.36	(0.3, 0.3)
c	79.00	(0.3, 0.3)
d	84.94	(0.1, 0.3)
e	90.51	(0.1, 0.3)
Grid cell 10x10		
a	48.00	(0.5, 0.6)
b	89.14	(0.1, 0.3)
c	77.42	(0.1:0.2, 0.3:05)
d	92.06	(0.1, 0.3)
e	92.68	(0.1, 0.3)

Table 4.1: Prior parameter values (p_{wall}, p_{space}) that result in maximum precision and the corresponding maximum precision values calculated and for each of the drawings in Figure 4.5.

(a variant of NSGA-II [44]) discussed in Section 4.4 was preformed for all five drawings. A (p_{wall}, p_{space}) value within the shortlisted Pareto optimal values for each drawing needs to be selected by examining the qualitative performance of different solutions. Tailoring the chosen prior parameters to the type of drawing used is impractical for real life applications and, ideally, a globally optimal value or range of values should be identified to ensure good results for any type of drawing.

Figure 4.5 shows the precision and recall interpolated colour maps for the five drawings used to generate the Pareto fronts in Figure 4.7. Figure 4.7 shows the Pareto optimal fronts [117] for all drawings and floor plans. There is a region where the Pareto optimal fronts for all drawings overlap, meaning there are optimal sets of (p_{wall}, p_{space}) values that yield good performance regardless of the type of drawing used. Only solutions that fall within set C as defined in Section 4.4 are of interest. These solutions need to be examined in a qualitative manner to determine which (p_{wall}, p_{space}) pair yield good performance for all drawings.

The subset of common Pareto optimal values that yield good qualitative performance is examined in the next section.

Grid cell 3x3		
Drawing	Max rec %	(p_{wall}, p_{space})
a	80.08	(1,0.2)
b	82.77	(0.2,0.1:0.2)
c	69.43	(0.1, 1)
d	82.70	(0.7:1, 1)
e	83.05	(0.5:1, 1)
Grid cell 5x5		
a	66.09	(1,1)
b	79.71	(0.2,0.2)
c	72.05	(1, 1)
d	86.00	(0.4:1, 1)
e	88.68	(0.9:1, 1)
Grid cell 10x10		
a	66.53	(1, 1)
b	73.42	(0.2, 1)
c	86.91	(1, 1)
d	91.69	(0.4:0.9, 1)
e	95.41	(0.2, 1)

Table 4.2: Prior parameter values (p_{wall}, p_{space}) that result in maximum recall and the corresponding maximum recall calculated for each of the drawings in Figure 4.5.

4.6.1 Identifying globally optimal prior parameters

A number of Pareto optimal (p_{wall}, p_{space}) pairs were identified for each drawing. Determining a different optimal pair for each drawing is impractical because, in a real life application, pre-testing the effects of different priors against a ground truth is not a realistic proposition. It is therefore important to determine values that yield good qualitative results that can be generalised across different drawings.

As shown in Figure 4.7 there are a number of Pareto optimal values that are common to all drawings. In order to further explore similarities, all Pareto optimal solutions can be viewed as a single dataset. These different values can then be analysed using a clustering algorithm to detect any clusters in the data. This will simplify the search for values that yield good qualitative results by allowing us to examine only a representative set of values from each cluster.

The k-means method [14, 71] was used in order to detect clusters in the data. This clustering algorithm is widely used and well documented, and a library performing k-means classification is available in Matlab.

The set of observations $\mathbf{P} = (p_1, p_2, ..., p_n)$, where p_j is the jth (p_{wall}, p_{space}) pair

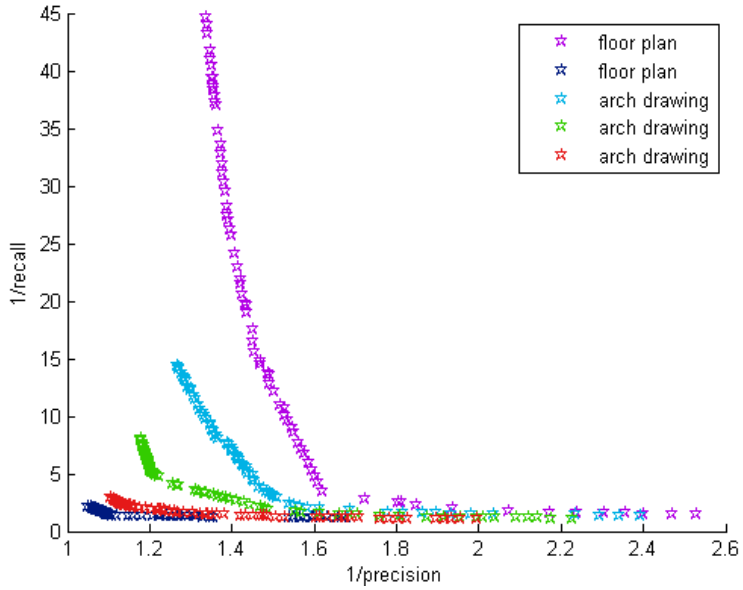


Figure 4.7: Pareto fronts for the representative floor plans and architectural drawings. There are a number of common solutions with precision and recall $> 40\%$, meaning there is an optimal set of (p_{wall}, p_{space}) that meets the requirements for all drawings.

and n the number of Pareto optimal pairs across all drawings is defined. Given this set, k-means aims to partition these observations into M clusters, with M being defined a priori. The assigned clusters are chosen to minimise

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{M} \sum_{\mathbf{P} \in S_i} \parallel \mathbf{P} - \mu_i \parallel^2 \tag{4.23}$$

where $\mathbf{S}$ is the set of cluster points, with S_i being the set of points for cluster i and μ_i the mean of the $i$the cluster.

Cluster means μ are initalised randomly. Given an initial set of means, $\mu_1, ..., \mu_M$ the algorithm alternates between the following steps until the cluster assignments no longer change and it converges.

The first step is often called the assignment step. At each time step k each observation is assigned to the cluster for which it minimises Equation 4.23

$$S_i^{(k)} = \{p_c : \parallel p_c - \mu_i^{(k)} \parallel^2 \leq \parallel p_c - \mu_j^{(k)} \parallel^2 \forall j, 1 \leq j \leq M\} \tag{4.24}$$

with each current obsevation, p_c being assigned to only one cluster, $S^{(k)}$, even if it meets the minimum distance requirement for more than one cluster.

The second step is the update step, in which new cluster means are computed

using

$$\mu_i^{k+1} = \frac{1}{|S_i^{(k)}|} \sum_{p_j \in S_i^{(k)}} p_j \tag{4.25}$$

The k-means variation used in this chapter is the Matlab implementation of k-means++ [14]. This variation uses an alternative method to initialise cluster means. The first mean μ_1 is selected at random from the observation set **P**. Then the distances between each remaining observation p_i and μ_1 are computed.

The next mean, μ_2 is selected at random from **P** with probability

$$\pi_2 = \frac{d^2(p_i, \mu_1)}{\sum_{j=1}^{n} d^2(p_j, \mu_1)} \tag{4.26}$$

where $d(p_i, \mu_1)$ is the distance between observation p_i and μ_1.

To determine the remaining means, μ_j, $j = 2, ..., M$ the distances from each observation to each mean are computed and each observation is assigned to its closest mean. Then for each observation p_i, $i = 1, ..., n$ the mean μ_j is selected at random form **P** with probability proportional to the distance between the μ_j and the mean closest to μ_j

$$\pi_j = \frac{d^2(p_i, \mu_l)}{\sum_{h:p_h \in \mathbf{C}_l}^{n} d^2(p_h, \mu_l)} \tag{4.27}$$

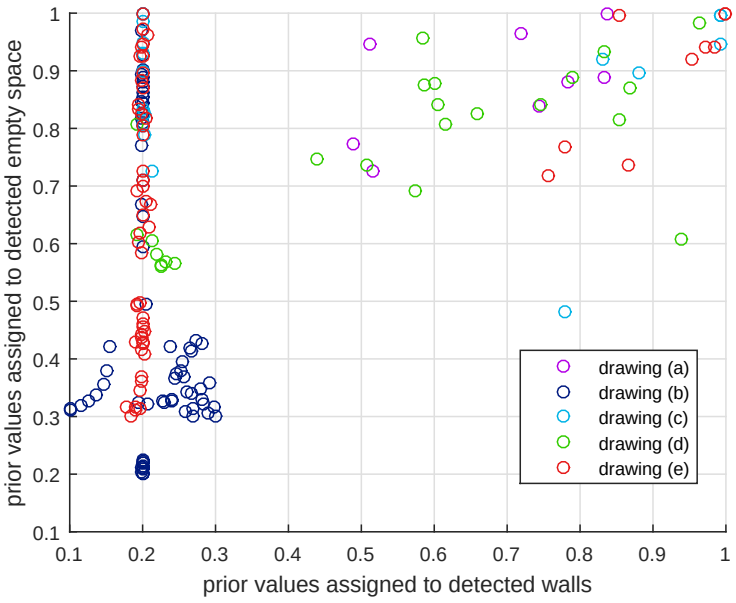
where $\mathbf{C}_l$ is the set of all observations closest to mean μ_l and $l = 1, ..., j-1$.

This process is repeated until all M means are chosen.

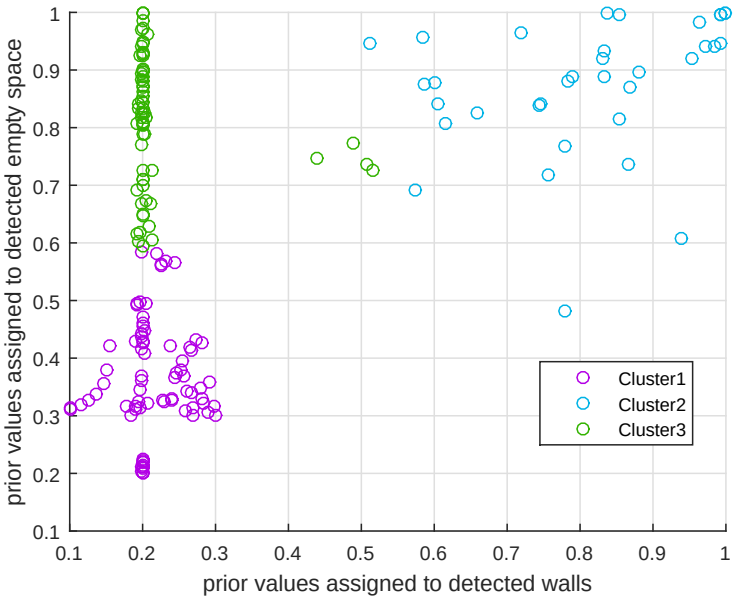
4.6.2 Optimised prior construction

In order to analyse the Pareto optimal values, all shortlisted (p_{wall}, p_{space}) pairs were plotted as shown in Figure 4.8(a). The k-means algorithm was then used to detect clusters within that data. The number of clusters was defined as $M = 3$. Values $2 \leq M \leq 4$ were also tested but the resulting clusters were either fragmenting the data too much for $M > 3$ or not separating the data enough for $M < 3$. The results of running the clustering algorithm are shown in Figure 4.8(b).

In order to identify an optimal cluster of solutions, representatives from each cluster were tested in terms of qualitative performance. Solutions that fall in Cluster 1, the purple cluster in Figure 4.8(b) have low values of both p_{wall} and p_{space} with $0.1 \leq p_{wall} \leq 0.3$ and $0.2 \leq p_{space} \leq 0.6$. The maximum precision solutions as shown in Table 4.1 lie within this cluster. Therefore this cluster of solutions tends to favour precision, resulting in maps that often contain very few correctly detected wall segments, yielding qualitatively poor results. Maps produced using (p_{wall}, p_{space}) pairs from Cluster 1 are shown in Figure 4.9, indicating poor map



(a)



(b)

Figure 4.8: Pareto optimal solutions for all drawings presented as a single data set. (a) Graph showing all the solutions presented in Figure 4.7 in one plot to examine where Pareto optimal solutions lie in the p_{space} vs p_{walls} plot, (b) Clusters detected within Pareto optimal solutions using the k-means clustering algorithm.

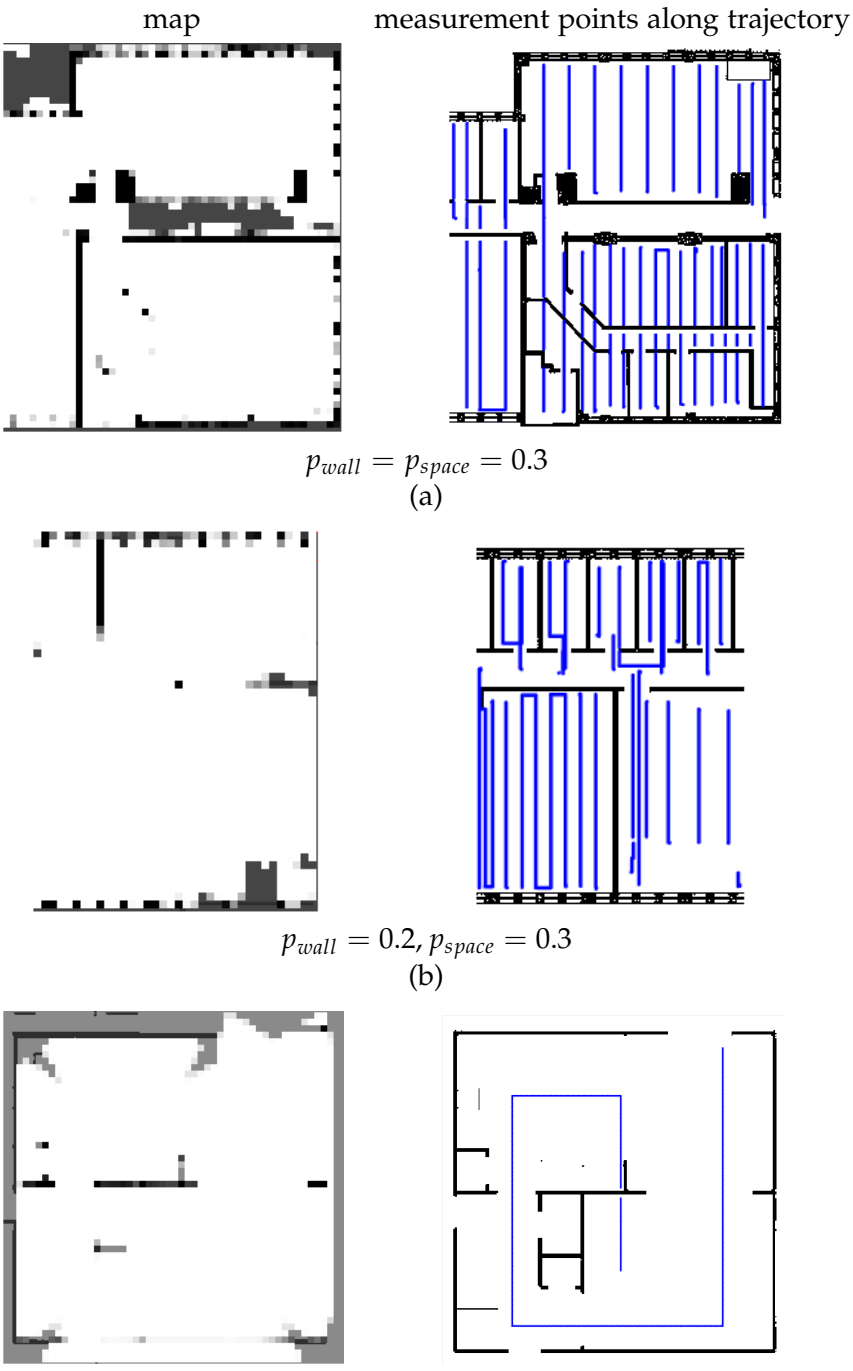


Figure 4.9: Final maps produced using values of (p_{wall}, p_{space}) from Cluster 1, the purple cluster in Figure 4.8(b), which correspond to high precision solutions for architectural drawings, (a) and (b), and for a floor plan, (c). The blue points in the right hand column correspond to the (x, y) points where measurements were taken; these do not correspond to complete trajectories since, thanks to the perfect knowledge of the robot's pose, we can use non-continuous measurement points.

quality.

Solutions in Cluster 2, the light blue cluster in Figure 4.8(b), have high values of both p_{wall} and p_{space}, with $0.5 \leq p_{wall} \leq 1$ and $0.48 \leq p_{space} \leq 1$. Solutions within this cluster correspond to the region where maximum recall solutions are observed, as shown in Table 4.2. These solutions tend to detect most of the building walls but also often incorrectly map empty space as occupied as shown in Figure 4.11.

Given the shortcomings of points in Clusters 1 and 2 the (p_{wall}, p_{space}) pair that yields the best qualitative performance lies in Cluster 3, the green cluster. The values in the green cluster are those that intuitively would be expected to perform well: a low p_{wall} and a high p_{space}, with $0.18 \leq p_{wall} \leq 0.21$ and $0.58 \leq p_{space} \leq 1$. In order to simplify the search, values with one decimal point accuracy were examined, with the possible solutions to investigate being the pairs

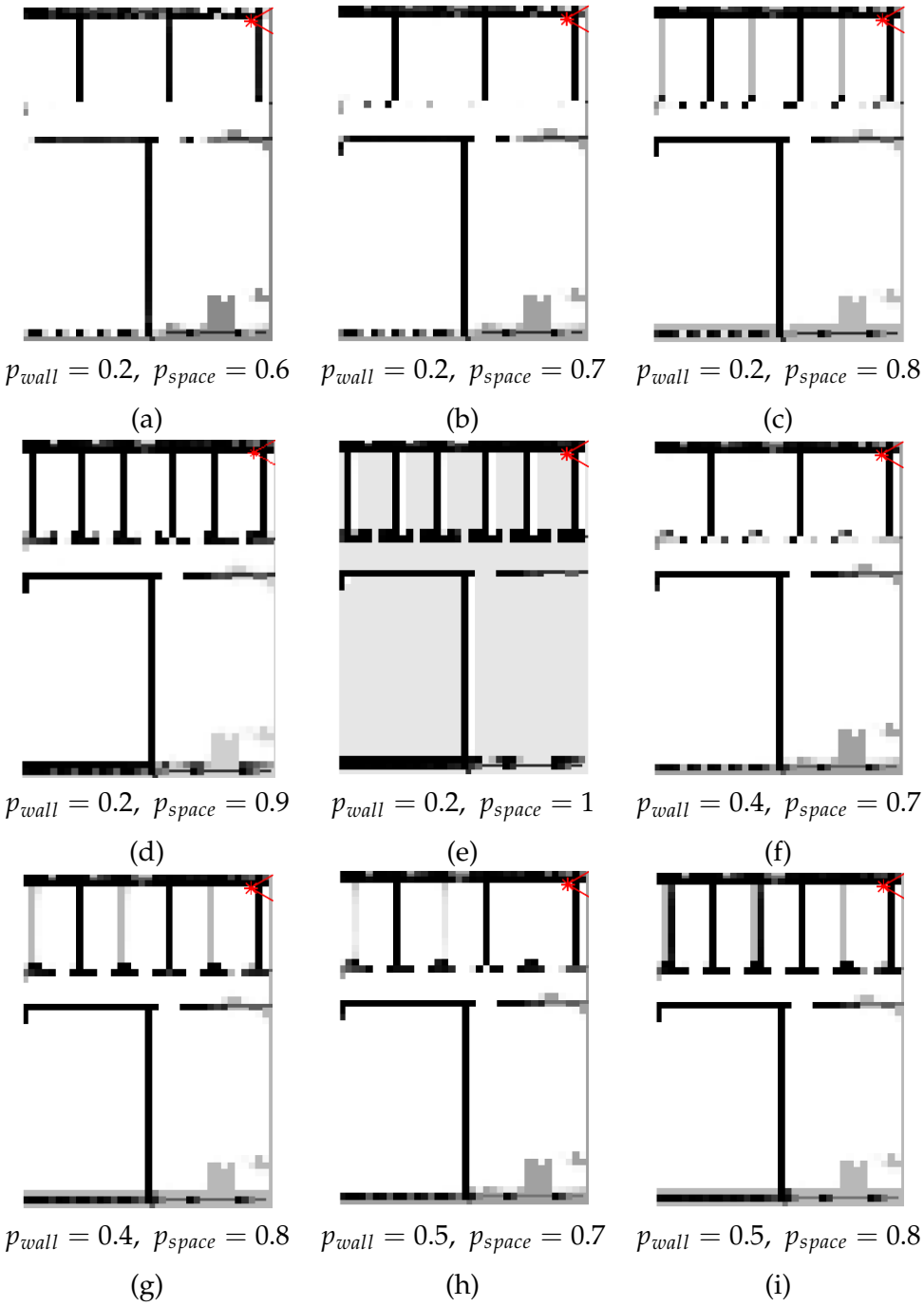
$$Shortlist = \begin{cases} p_{wall} = 0.2,\ 0.6 \leq p_{space} \leq 1 \\ 0.4 \leq p_{wall} \leq 0.5,\ 0.7 \leq p_{space} \leq 0.8 \end{cases} \tag{4.28}$$

The maps produced using each of these pairs were compared in terms of visual quality and the best (p_{wall}, p_{space}) combination out of the green cluster points in terms of qualitative performance was found to be $p_{wall} = 0.2$, $p_{space} = 0.9$. These results lead to the conclusion that using a prior is beneficial and starting values of $p_{wall} = 0.2$, $p_{space} = 0.9$ yield good qualitative results. The following chapter conducts a qualitative and quantitative comparison of maps produced using this proposed informative prior and an the commonly used non-informative prior.

4.7 Concluding remarks

This chapter presented a method to shortlist Pareto optimal values to assign to indoors occupancy grid mapping priors. Priors were constructed using structural information extracted from architectural drawings and floor plans by assigning appropriate prior probabilities to detected wall and empty space locations. A precision-recall analysis was used to assess quantitative performance and a multi-objective optimisation was used to shortlist Pareto optimal solutions. Both types of drawings were found to have a similar region of (p_{wall}, p_{space}) that yields good performance metrics. A method to construct such informative Bayesian priors for any drawing by processing the Pareto optimal values shortlisted in this chapter to identify a (p_{wall}, p_{space}) pair that yields good performance across all drawings was also presented.

The next chapter benchmarks the maps produced using the proposed informa-

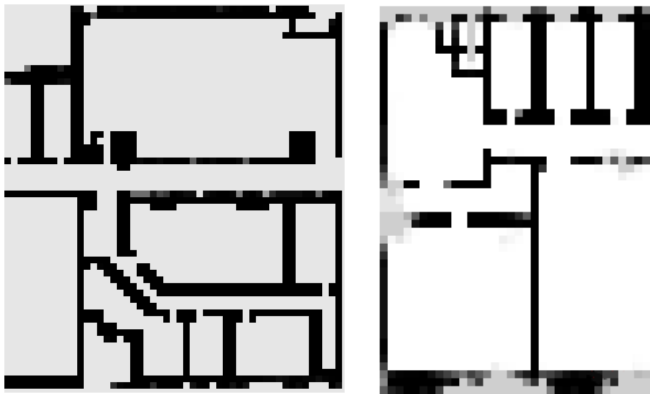


$p_{wall} = 0.2,\ p_{space} = 0.6$ (a) $p_{wall} = 0.2,\ p_{space} = 0.7$ (b) $p_{wall} = 0.2,\ p_{space} = 0.8$ (c)

$p_{wall} = 0.2,\ p_{space} = 0.9$ (d) $p_{wall} = 0.2,\ p_{space} = 1$ (e) $p_{wall} = 0.4,\ p_{space} = 0.7$ (f)

$p_{wall} = 0.4,\ p_{space} = 0.8$ (g) $p_{wall} = 0.5,\ p_{space} = 0.7$ (h) $p_{wall} = 0.5,\ p_{space} = 0.8$ (i)

Figure 4.10: Maps produced using the different Pareto optimal $(p_{wall},\ p_{space})$ pairs found in the green cluster of Figure 4.8(b), with the map shown in (d) yielding the best visual results, with $p_{wall} = 0.2,\ p_{space} = 0.9$ being the proposed informative prior values.



$p_{wall} = p_{space} = 1$ $p_{wall} = p_{space} = 0.9$

(a) (b)

Figure 4.11: Final maps produced using values of (p_{wall}, p_{space}) from Cluster 2, the light blue cluster in Figure 4.8(b), which correspond to high recall solutions, for two architectural drawings, (a) and (b).

tive prior against those produced using a non-informative prior both in simulation and using experimental results.

Chapter 5

Constructing and using optimised informative priors

The use of a non-informative prior is common practice in the occupancy grid SLAM literature, with most researchers using such a prior in order to produce solutions that do not depend on having prior knowledge of the environment [50]. There have thus been no studies of the benefits of using an informative prior. This chapter benchmarks the globally optimal prior values, (p_{wall}, p_{space}), against the non-informative prior. Maps produced using the proposed informative prior are compared to those constructed using a non-informative prior in both a quantitative and a qualitative sense. A summary of the simulation and experimental results presented in this chapter is published in the International Journal of Robotics Research [60].

The contributions of this chapter are as follows:

- A qualitative comparison of maps produced using a non-informative prior and those produced using the proposed informative prior
- A quantitative benchmarking of maps produced using a non-informative prior and those produced using the proposed informative prior
- A presentation of the benefits of using an informative prior in challenging real life scenarios
- An experimental validation of simulation results

This chapter is organised as follows. Section 5.1 discusses how the benefits of using an informative over a non-informative prior are assessed. Section 5.2 presents qualitative and quantitative results of this comparison, with the proposed informative prior yielding an increase in the F_2 metric of at least 20%. Section 5.3

shows experimental results using real sensor data collected using a Kinect and processed using the proposed Matlab code, highlightling areas where using a prior can yield improved performance.

5.1 Assessing the benefits of using informative priors over uninformative ones

Once a (p_{wall}, p_{space}) pair that performs well for all drawings has been identified, the benefits of using this proposed informative prior need to be studied. In order to compare the performance of using the informative and the non-informative priors in a qualitative sense, maps produced using each type of prior are compared for the following cases:

- Different grid resolutions
- Different drawings
- Thorough as well as quick exploration of the environment

Assessing performance quantitatively is more challenging given the conflicting goals of optimising precision and recall. Given the multi-objective nature of the optimisation, merely comparing the values of precision and recall for maps produced using the informative and non-informative prior does not provide enough information. In order to overcome this problem the F_β metric [128] is used to allow a comparison of the precision and recall combination rather than individual values for each drawing

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{(\beta^2 \times precision) + recall} \tag{5.1}$$

where β determines whether precision or recall is favoured. For applications where a high recall is considered to be more important than high precision $\beta > 1$ is chosen and $\beta < 1$ is chosen if precision is favoured over recall. The $F_{0.5}$, F_1 and F_2 measures are commonly used. The F_1 measure is the harmonic mean of precision and recall, $F_{0.5}$ favours precision over recall and F_2 favours recall over precision. Measures for $\beta < 0.5$ or $\beta > 2$ are less common since they greatly favour one metric over the other.

In this thesis the F_2 measure is used

$$F_2 = 5 \times \frac{precision \times recall}{(4 \times precision) + recall} \tag{5.2}$$

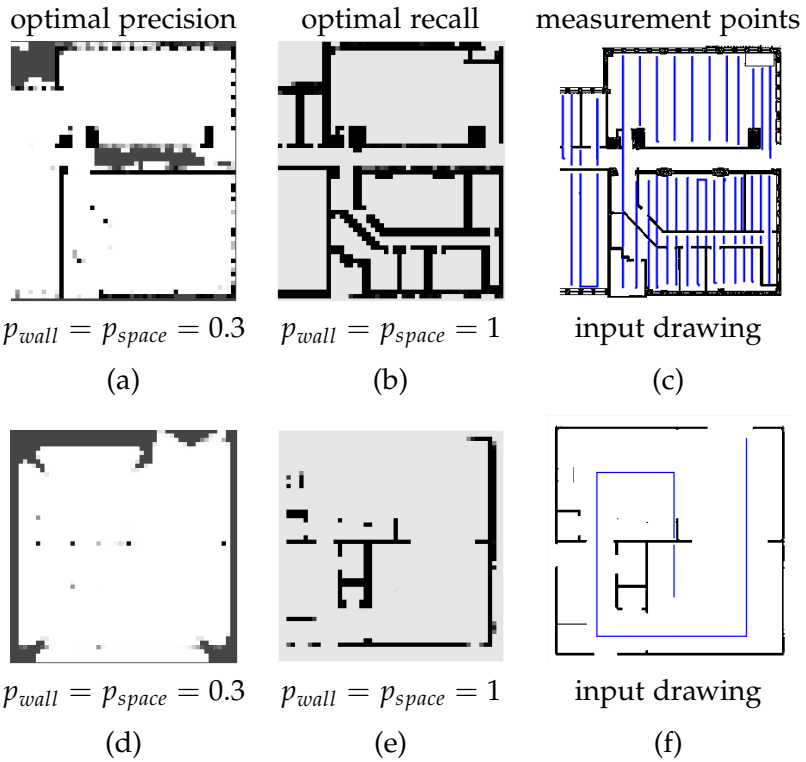


Figure 5.1: Final maps produced using values of (p_{wall}, p_{space}) that optimise precision (a), (d) and values that optimise recall (b), (e) for an architectural drawing, (a), (b) and a floor plan, (d), (e). The blue points in the right hand column correspond to the (x, y) points where measurements were taken; these do not correspond to complete trajectories since, thanks to the perfect knowledge of the robot's pose, we can use non-continuous measurement points.

This measure favours recall over precision, Equation 5.2. Recall is favoured over precision since it represents the percentage of correctly detected walls and thus affects the map outline more, Figure 5.1.

5.2 Simulation results and discussion

The maps produced using the proposed informative prior and a non-informative prior are shown in this section. The two approaches are also compared in a quantitative sense using the F_2 metric. A study is also conducted to examine how the quality of the prior map affects the quality of the final map produced. The benefits of using a prior map if a quick exploration of the environment is required are also presented. Finally, the benefits of using an informative prior over methods that do not use probabilistic recursive map estimation are presented.

The trajectory poses used to update the map are presented in each case, show-

ing the locations at which sensor measurements were captured. These are not always complete trajectories, since enough measurements to map the majority of the environment were used. Given our simulator only models $\theta = [0,\ \frac{\pi}{2},\ \pi,\ \frac{3\pi}{2}]$, whenever the robot moves in a vertical line in the map, alternating sensor measurements at $\theta = 0$ and $\theta = \pi$ are taken as the robot moves. Similarly, whenever the robot moves in horizontal lines in the map alternating measurements at $\theta = \frac{\pi}{2}$ and $\theta = \frac{3\pi}{2}$ are taken.

In more detail, the angle θ at which measurements are taken is determined as follows. When the robot moves in a vertical line in the image, $x_i = x_{i+1}$

$$\theta = \begin{cases} \theta = 0,\ for\ \ i \ \mathbf{mod}\ 2 = 0 \\ \pi,\ for\ \ i \ \mathbf{mod}\ 2 = 1 \end{cases} \tag{5.3}$$

when the robot moves in a horizontal line along the image, $y_i = y_{i+1}$

$$\theta = \begin{cases} \pi/2,\ for\ \ i \ \mathbf{mod}\ 2 = 0 \\ 3\pi/2,\ for\ \ i \ \mathbf{mod}\ 2 = 1 \end{cases} \tag{5.4}$$

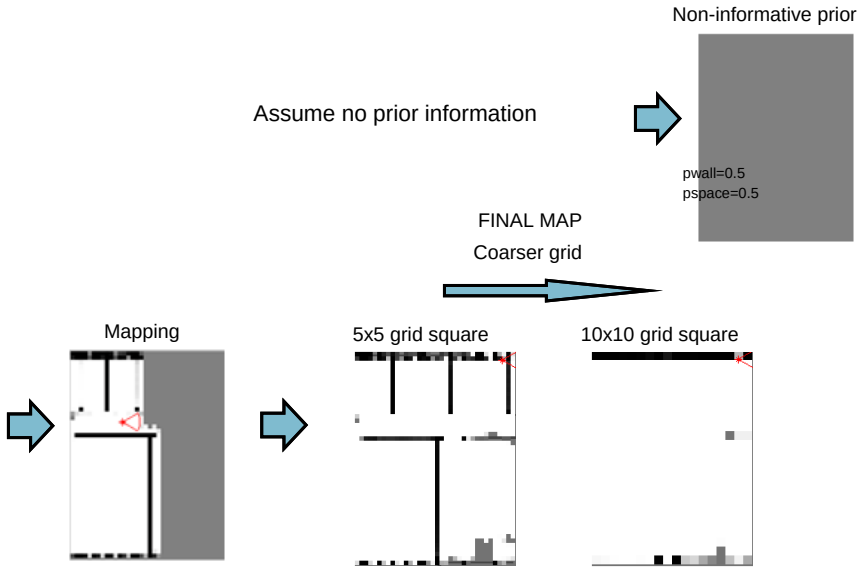
for each time step i.

5.2.1 Qualitative comparison

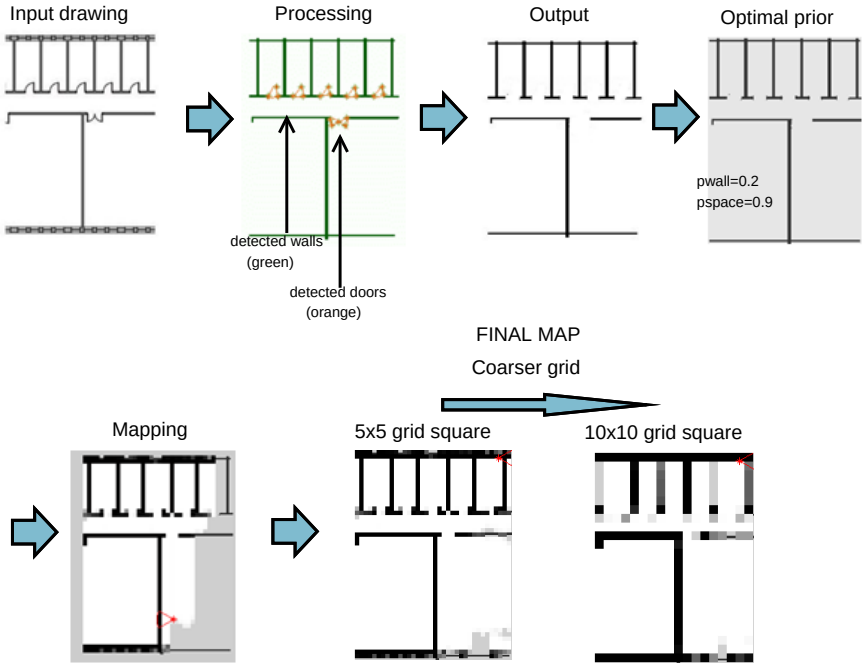
An overview of the different priors and resulting maps using the same exploration path for two different grid resolutions are shown in Figure 5.2. Using a non-informative prior is found to produce maps that miss building walls, especially when a coarse grid is used. Conversely, using an informative prior yields maps of better quality, not missing any wall sections, even for a coarser grid. Results for the remaining drawings for three different grid resolutions are presented in Figures 5.3-5.6. The points along the robot trajectory used to update the map are shown in Figures 5.3-5.6(d). The area covered by the sensors can be deduced by looking at the left hand column of Figures 5.3-5.6, in which unexplored areas are shown in grey.

Using the proposed informative prior yields improved maps for all drawings, detecting the majority of wall sections even when coarser grids are used. Therefore if computationally cheaper mapping is required due to limited available processing power a coarser grid and informative prior can be used.

This optimised prior also allows for improved performance when a robot performs a quick and/or incomplete exploration of the environment, leading to un-



(a) Commonly used method: using non-informative prior



(b) Proposed method: using proposed informative prior

Figure 5.2: Comparison of map produced using the proposed informative prior and that produced using the non-informative prior. (a) The process of processing and converting a drawing to an optimised prior and the map produced using this prior for two different grid resolutions, (b) The non-informative prior and the map produced using it for two different resolutions.

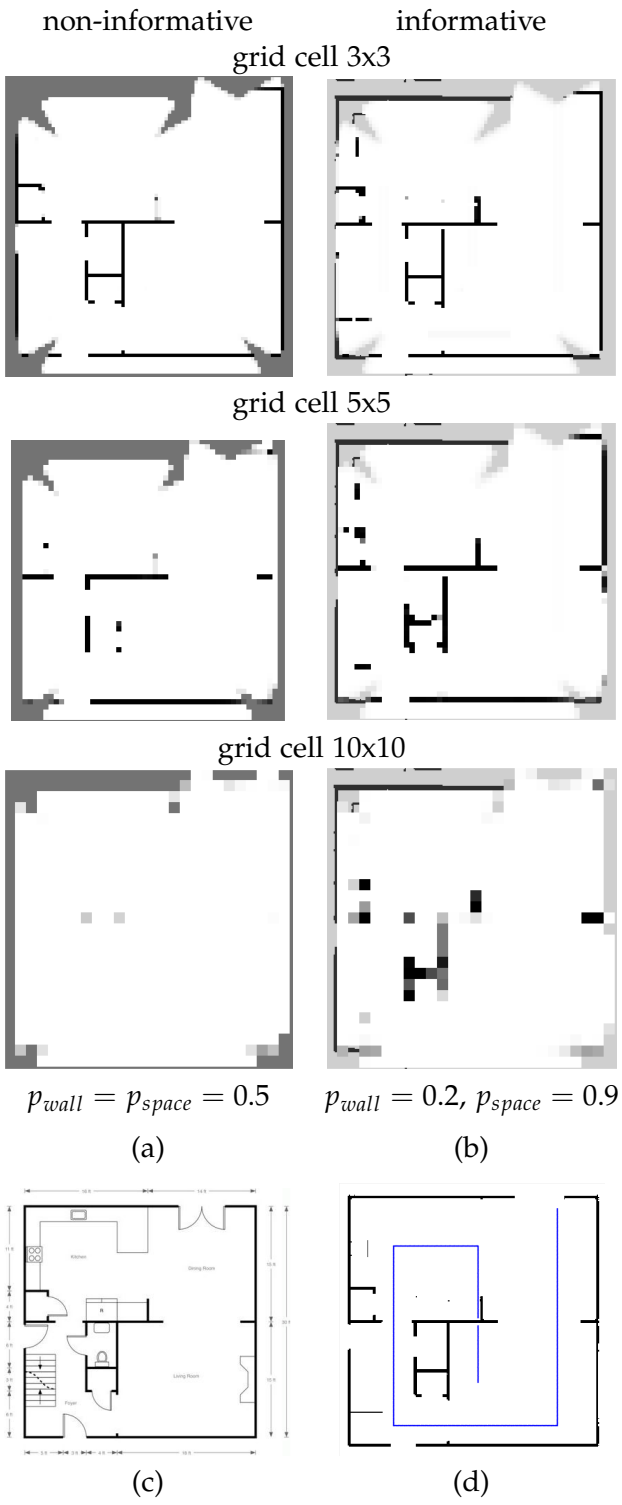


Figure 5.3: Final maps produced using a non-informative prior (column (a)) and using the proposed informative prior (column (b)) for three different grid resolutions, with the input drawing shown in (c) using measurements taken at positions shown as blue points in (d).

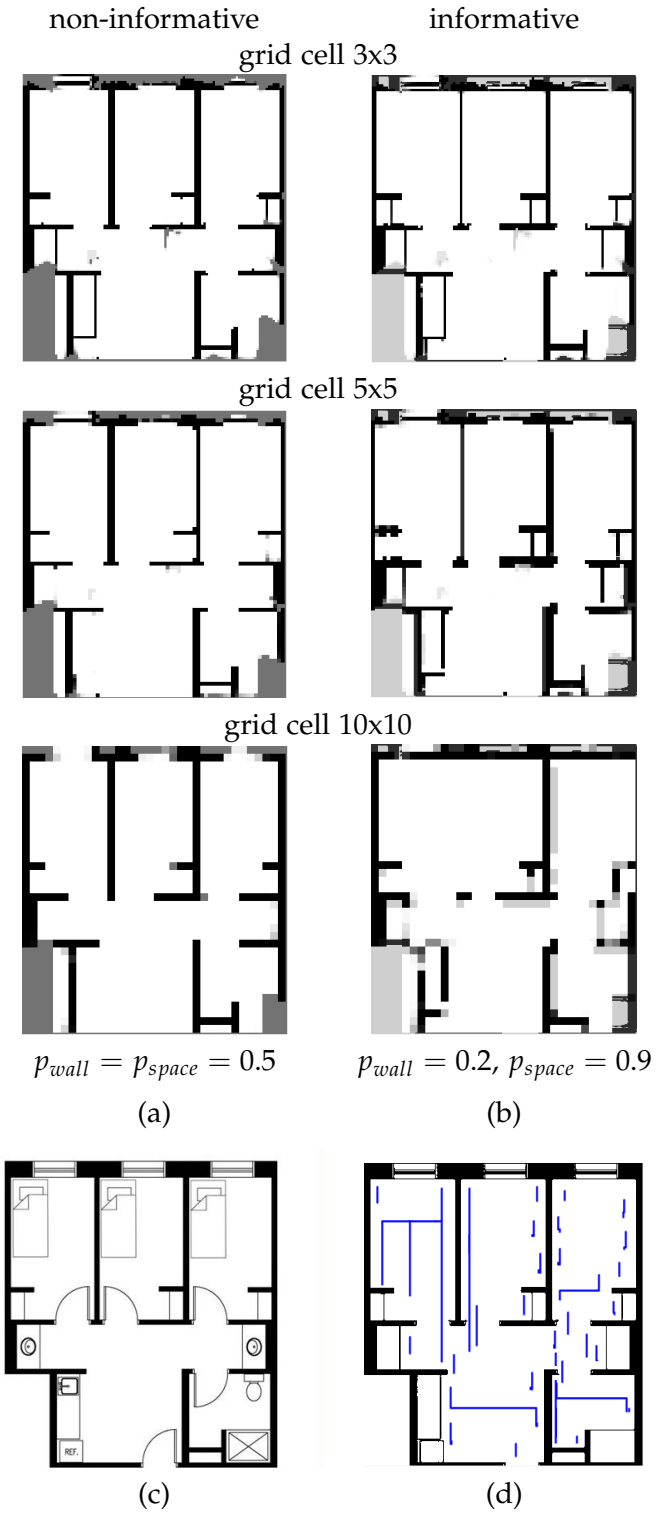


Figure 5.4: Final maps produced using a non-informative prior (column (a)) and using the proposed informative prior (column (b)) for three different grid resolutions, with the input drawing shown in (c) using measurements taken at positions shown as blue points in (d).

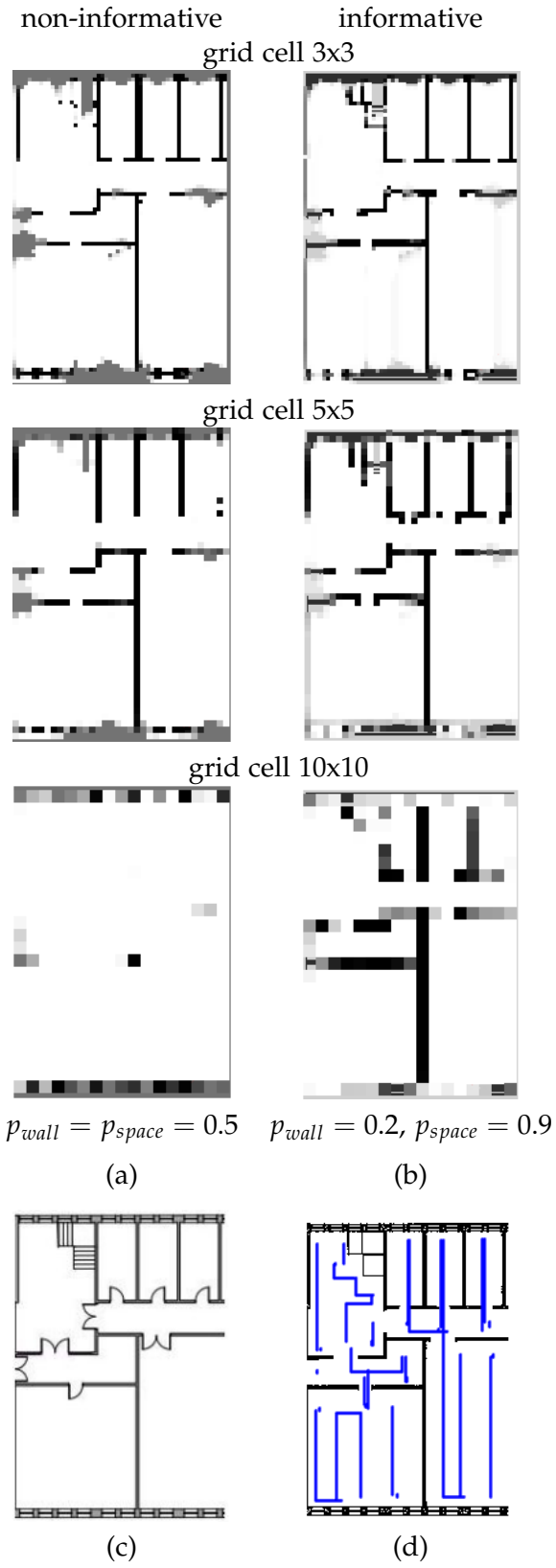


Figure 5.5: Final maps produced using a non-informative prior (column (a)) and using the proposed informative prior (column (b)) for three different grid resolutions, with the input drawing shown in (c) using measurements taken at positions shown as blue points in (d).

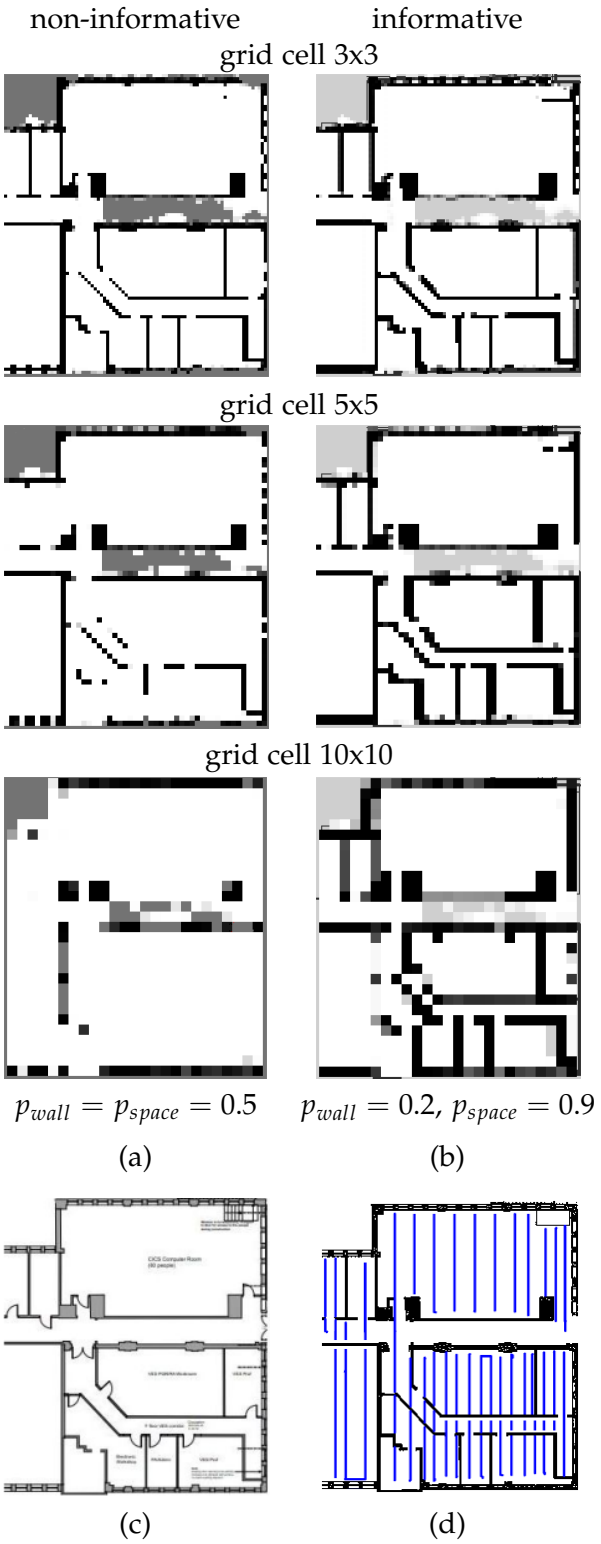
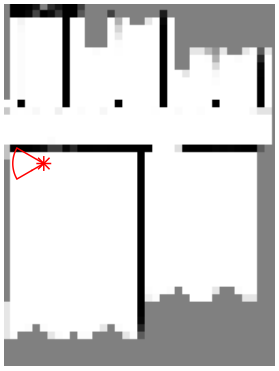


Figure 5.6: Final maps produced using a non-informative prior (column (a)) and using the proposed informative prior (column (b)) for three different grid resolutions, with the input drawing shown in (c) using measurements taken at positions shown as blue points in (d).

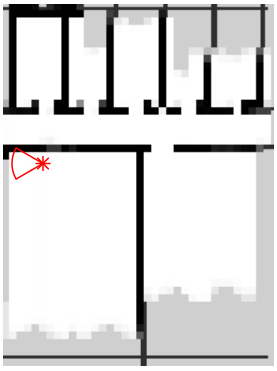
non-informative



$p_{wall} = p_{space} = 0.5$

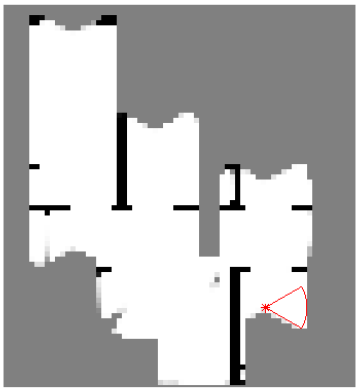
(a)

informative

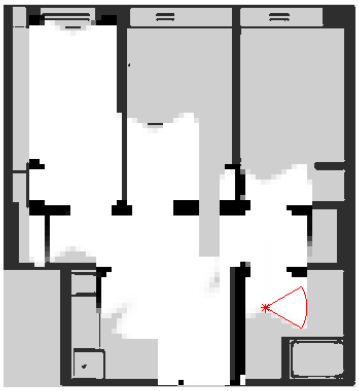


$p_{wall} = 0.2$, $p_{space} = 0.9$

(b)



(c)



(d)

Figure 5.7: Final maps produced using a non-informative prior, (a) and (c) and using the proposed informative prior, (b) and (d) for an incomplete exploration of the environment using an architectural drawing ((a), (b)) and a floor plan ((c), (d)) to extract prior information.

explored areas or areas scanned only once or twice, making the effect of the prior more significant. This type of exploration may be required if a robot is deployed in a time-critical mission such as USAR. In the mapping example presented in Figures 5.7(a) and (b) the robot has not fully explored the environment. In such a case using an informative prior has the added benefit of providing information about building sections even if areas are unexplored, with the non-informative prior, Figure 5.7(a), yielding a significantly worse map than the informative prior Figure 5.7(b).

In the mapping example shown in Figures 5.7(c) and (d) the robot has explored one room fully but has obtained few scans for the remaining rooms. Using a prior provides information about remaining rooms and overall building structure and helps make sense of sensor readings, Figure 5.7(d). The map produced using the non-informative prior, Figure 5.7(c), provides very little information about the building and is difficult to interpret. The number of rooms and overall building structure cannot be deduced, making the map ill-suited for use in safety critical missions.

Mathematically, this superior performance can be attributed to the fact that when each cell is scanned by the distance sensor only a few times or not at all the prior plays an important role. Looking at the grid cell update as described in the occupancy grid mapping algorithm presented in Chapter 4 the log odds of occupancy of each grid cell i is updated using

$$l_{k,i} = InverseSensorModel(\mathbf{m}_i, \mathbf{x}_k, \mathbf{z}_k) + l_{k-1,i} - l_{0,i} \tag{5.5}$$

At time $k = 0$, $l_{k,i} = l_{0,i}$. If a certain square is not scanned by the sensor the log odds of occupancy for that square is its prior value $l_{0,i}$. If only one or a small number of scans are performed then the effect of the prior is more significant. Therefore if a quick and computationally cheaper exploration is required the proposed optimised prior yields significantly better maps in terms of visual quality, identifying the majority of building walls.

The overall approach proposed in this thesis to process an architectural drawing or floor plan, extract structural information and use it to construct an optimised prior that can be used to produce accurate recursively updated maps is shown in Figure 5.2.

5.2.2 Quantitative comparison

The F_2 metric was calculated for maps produced using an informative and uninformative prior for all drawings. As shown in Figure 5.8 using an informative prior

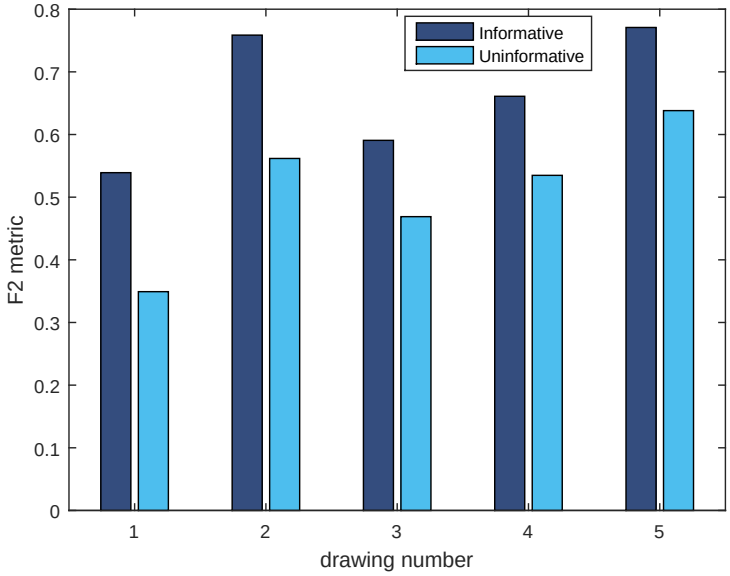


Figure 5.8: Quantitative comparison of the maps produced using an informative and a non-informative prior using the F_2 metric.

yields an increase in the F_2 measure of at least 20% over the uninformative prior for all five drawings. It is also worth examining the importance of the prior as the robot explores the environment, shown in Figure 5.9 for the drawing of Figure 5.2. The red line depicts the evolution of the F_2 metric for the map produced using the proposed informative prior from time $k = 0$ until the end of the exploration. The blue line shows the same for the map produced using the non-informative prior.

This figure further highlights the advantages of using an informative prior: if a quick exploration is required, using an informative prior can produce a map with a higher F_2, while the map produced using the non-informative prior may never reach a high F_2 value. Given that realistic SLAM applications such as USAR or commercial robots need to be able to explore their environment quickly and efficiently, using an informative prior can help produce better quality maps faster, without adding to the computational cost of running the SLAM algorithm.

5.2.3 Effects of prior map quality on final map

While benchmarking the proposed informative prior against the non-informative prior it is worth examining how the quality of the extracted prior affects performance and whether it is still worth using a prior even if the quality of extracted information is poor. In order to test this, different qualities of priors were tested. An example is presented in Figure 5.10, showing the maps produced in the right

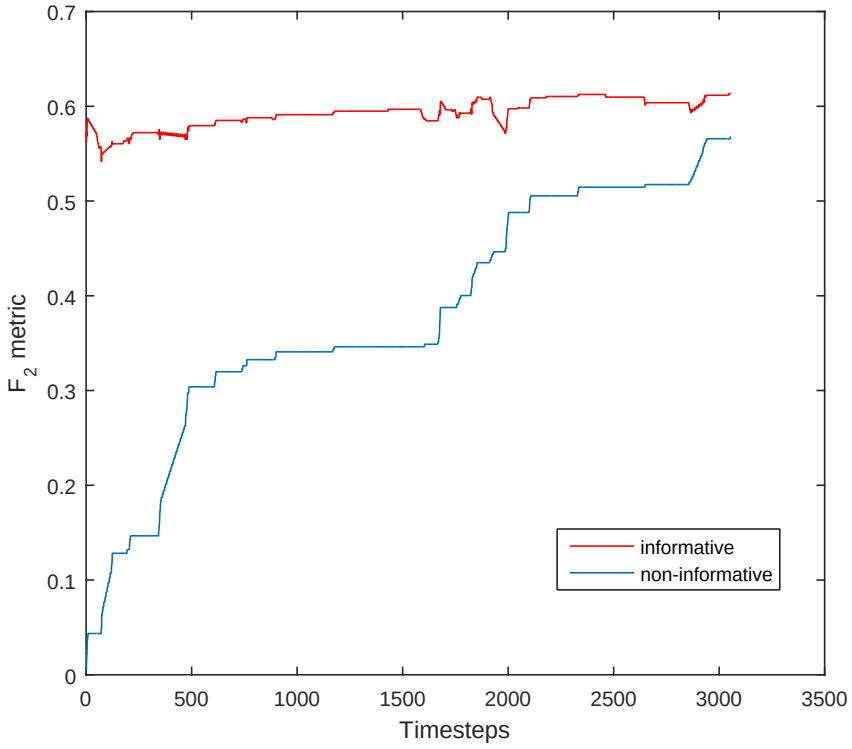


Figure 5.9: A comparison of the F_2 metric at each time step as the environment is explored using an informative and a non-informative prior for the drawing in Figure 5.2. The map produced using an informative prior starts at a high F_2 value and maintains a high F_2 metric throughout.

hand column using the priors in the left column. The first two priors (a) and (c) are incomplete, missing large parts of the building structure. The prior in (e) contains an extra wall that is not present in the environment. In all these cases the map produced is superior in terms of qualitative performance to the map produced using the uninformative prior.

This result also indicates that, even if the algorithm used to extract prior information does not perform well or if sections of the environment are altered, the proposed method still performs well, yielding an improvement over the non-informative prior.

5.3 Experimental results

In order to experimentally validate the results obtained in this chapter and highlight potential problems of real sensors, simulated sensor data were substituted for data collected using a Kinect sensor mounted on a turtlebot [4], Figure 5.11. These

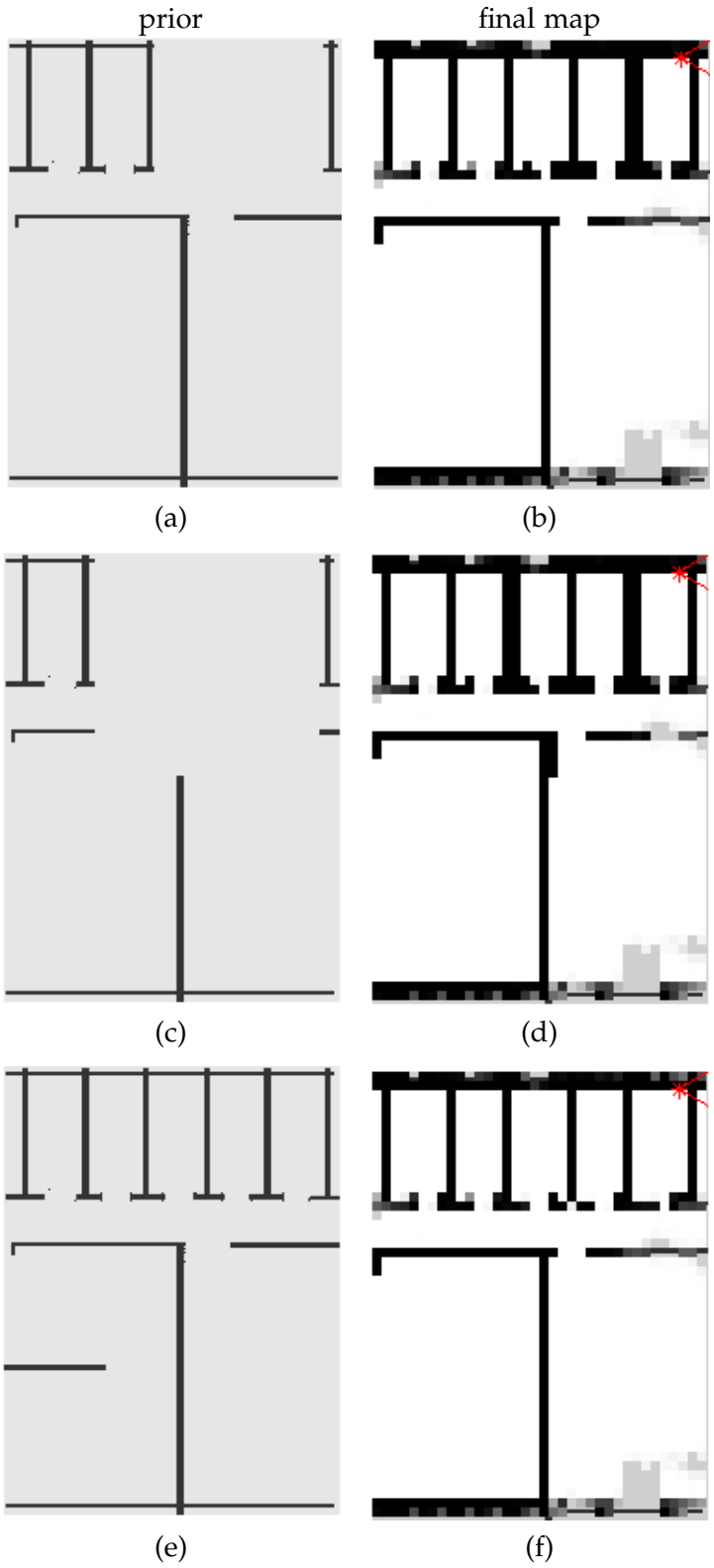


Figure 5.10: Final maps produced using incorrect or incomplete prior maps. (a) A prior missing building sections, (c) a prior that is missing a large number of building walls and (e) a map that contains a wall that is not present in the environment and (b), (d) and (f) maps produced using the priors (a),(c) and (e) respectively.

results bring the majority of situations discussed in this chapter together, demonstrating areas in which sensor readings are unreliable, an incomplete exploration or a limited number of sensor readings.

The overarching assumption that the robot pose is known was maintained and the simulator presented in Section 4.5.2 was used to analyse real Kinect data. The Kinect's field of view is a cone with a horizontal field of view of 58°, a minimum range of 40 cm and a maximum range of 3 m [5, 6].

The Kinect sensor yields 480×640 pixel distance images, with each image pixel being assigned a distance value. These had to be converted to 2D distance readings, as required for 2D mapping. This conversion was performed by averaging distance values along a 20×640 pixel strip across the image, at a height determined by the height of walls in the environment, Figure 5.12. This averaging was performed to avoid calculating a range estimate based on a very thin strip, which could result in a small sensor error yielding a large 2D measurement error. This method approximates a 2D sensor that is not always accurate, with accuracy depending on its distance from different wall heights and the quality of distance data across the distance image. This model allows for a study of the benefits of using a prior when a lower quality sensor is used.

The possible pose angles modeled in the simulator, θ, are the same as for the simulated data, only modeling angles $\theta = [0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}]$. This simplification allows us to explore the potential benefits of using a prior when a complete exploration of the environment would be computationally expensive.

A simulated building with three rooms was used as the environment to be explored, using a Sheffield Robotics arena with dimensions 3×4 m, Figure 5.11(a). Three rooms were modeled because, due to the minimum range of the Kinect sensor, a more complex environment would require a larger arena. Prior information extracted from a drawing was assumed to be very accurate, as shown in Figure 5.13(a). The trajectory followed is show in Figure 5.13(b), highlighting points where distance measurements were taken (in blue) and the measurements (in red).

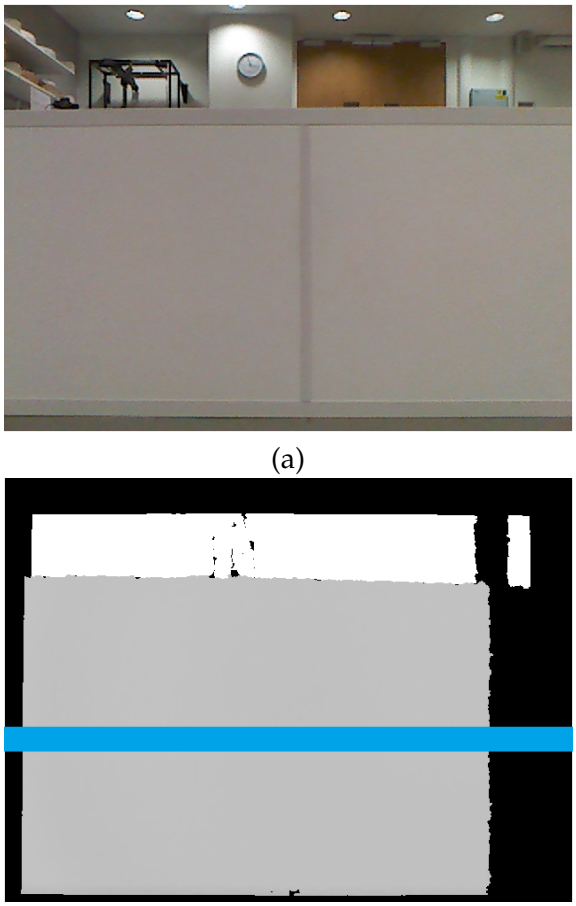
The (x, y) position of the turtlebot in the course was measured by hand using a measuring tape. The heading θ was also determined by hand, by setting the turtlebot such that the Kinect sensor was parallel to one of the course walls at all times, making it easier to set up (a rectangular object such as a box was used to ensure the turtlebot was parallel to the course walls for each measurement point). Using this method we recorded the robot's (x, y, θ) pose for each measurement point, along with the corresponding distance measurements, z.

The multi-objective optimisation proposed in Section 4.4 was repeated using

(a)

(b)

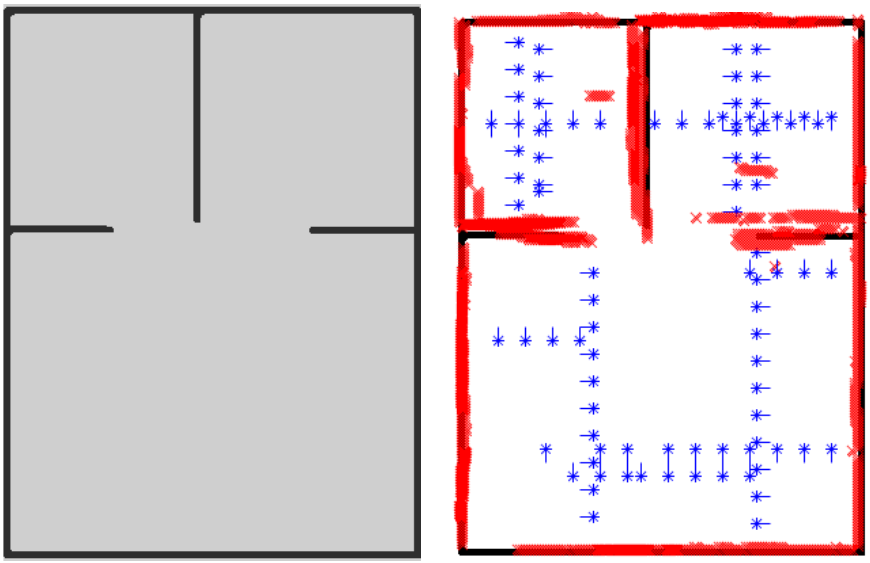
Figure 5.11: The experimental setup used to obtain real sensor readings: a turtlebot running ROS and mounted with a Kinect sensor was used to collect distance readings at known robot poses. (a) 4×3 m simulated building, with three rooms; (b) turlebot mounted wth Kinect sensor.



(a)

(b)

Figure 5.12: An example of a photo and corresponding distance image obtained using the Kinect sensor, (a) RGB image and (b) corresponding distance image. Darker pixels in (b) correspond to objects nearby and lighter pixels correspond to objects further away. The blue strip overlaid on the distance image shows the strip of pixels used to perform averaging to obtain 2D distance readings.



(a) (b)

Figure 5.13: (a) The accurate prior map of the building, assuming walls were detected correctly. The proposed informative prior values were assigned to walls and empty space, with $p_{walls} = 0.2$ and $p_{space} = 0.9$; (b) Trajectory points where measurements were taken shown as blue stars with robot heading indicated by the blue line, obstacles detected shown in red.

the real Kinect data collected during the experiment. The following model was used, based on [127]

$$InverseSensorModel(\mathbf{m}_i, \mathbf{x}_k, \mathbf{z}_k) = \begin{cases} 0.9, & n_i^{pixels<z} = n_i^{pixels} \\ 0.1, & n_i^{pixels>z} = n_i^{pixels} \\ \dfrac{n_i^{pixels<z}}{n_i^{pixels}}, & 0 < n_i^{pixels<z} < n_i^{pixels} \end{cases} \tag{5.6}$$

with $n_i^{pixels<z}$ being the number of map pixels for a cell i that fall between the robot and the location of the detected obstacle, z; $n_i^{pixels>z}$ the number of map pixels for a cell i that are located beyond the detected object or at the location of the detected object; and n_i^{pixels} the number of map pixels in cell i. For an accurate sensor, cells for which all map pixels correspond to detected empty space are assigned a value of 0.9, those that correspond to detected objects a value of 0.1 and those that contain both empty space and objects are assigned a value equal to the percentage of pixels corresponding to detected objects.

A very narrow range of optimal (p_{wall}, p_{space}) was observed, which mostly agree with the results obtained in simulation, yielding opimised values of $p_{wall} = 0.2$ and $p_{space} = 1$. These values are multiplied by a factor of 0.9 before they are used in the prior map to avoid values too close to 1. A less accurate sensor model assigning 0.8 to detected empty space and 0.2 to detected walls was also tested, yielding optimised values $p_{wall} = 0.4$ and $p_{space} = 0.75$.

Final maps produced using the non-informative and the proposed informative prior for two different grid resolutions are shown in Figure 5.14 for an exploration that collects 3 sensor readings for each robot pose. The maps produced using the non-informative prior and the informative prior for an exploration that collects only 1 sensor reading for each pose are shown in Figure 5.18.

When a finer grid is used, Figure 5.14(a), using a prior makes less of a difference. However, it still yields improved performance in terms of detecting outer walls such as the right and bottom outer walls. When a coarser grid is used, Figure 5.14(b), the proposed prior results in some of the outer and inner walls being detected, whereas the map produced using the non-informative prior effectively provides no information. Areas where using the informative prior yields visibly better performance, areas 1-4 in Figure 5.15, are shown in Figures 5.16 and 5.17. Finally, neither approach can correct well for erroneous sensor readings, such as the blocky inner wall and non-existent obstacles detected in the the top right room. This further highlights potential benefits of using the prior: if an informative prior is used areas that are very different from what was expected in the prior can be tested further to determine whether they correspond to errors, clutter or collapse, in the case of USAR.

If a quick exploration is performed, using only a single sensor reading to update the map for each pose, the proposed prior helps detect outer walls more accurately, Figure 5.18(a) and Figure 5.18(b). In areas where there are sensor readings that are overlapping, such as the inner middle wall and the bottom half of the left outer wall, the difference between the two maps is smaller as would be expected. When a coarser grid is used, Figure 5.18(c) and Figure 5.18(d), the proposed prior yields a map that has correctly identified some of the inner and outer walls despite the coarser grid used and the single sensor reading used to update the map, Figure 5.18(d). Conversely, the map produced using the non-informative prior has failed to identify any internal or outer walls, Figure 5.18(c). Areas where using the informative prior yields visibly better performance, areas 1-4 in Figure 5.15, are shown in Figures 5.19 and 5.20.

The evolution of the F_2 metric with time is plotted for the map produced using the informative prior and that produced using the non-informative prior for a

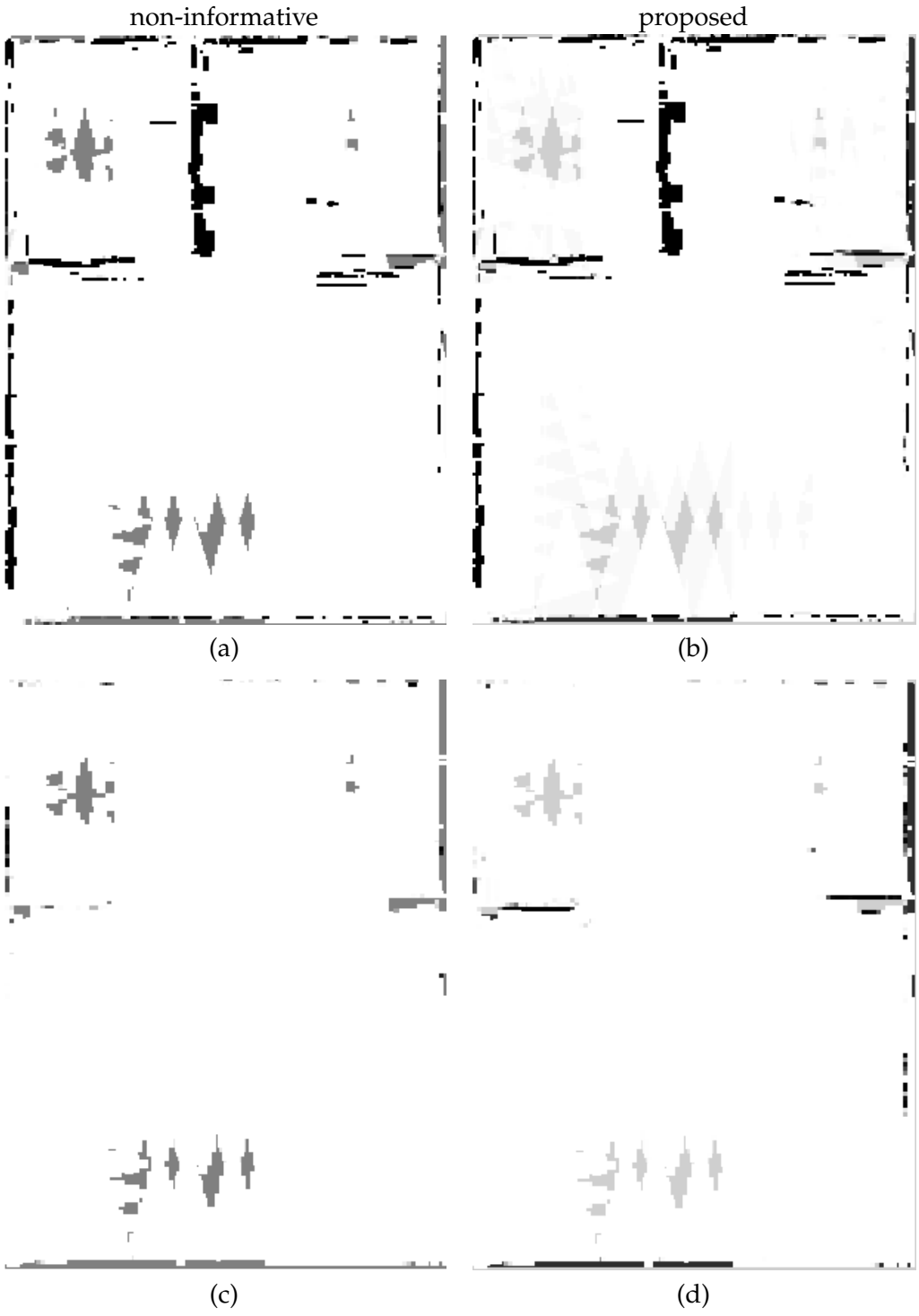


Figure 5.14: Final maps produced using the non-informative prior, (a), (c) and maps produced using the proposed informative prior, (b), (d), using 3 sensor readings per robot pose. Maps (a) and (b) were produced using a finer grid, with a grid cell side of 2, and (c) and (d) were produced using a coarser grid, with a grid cell side of 3. In maps (a) and (c) grey areas such as the top right and bottom left in (c), correspond to a probability of 0.5 providing no information about the environment.

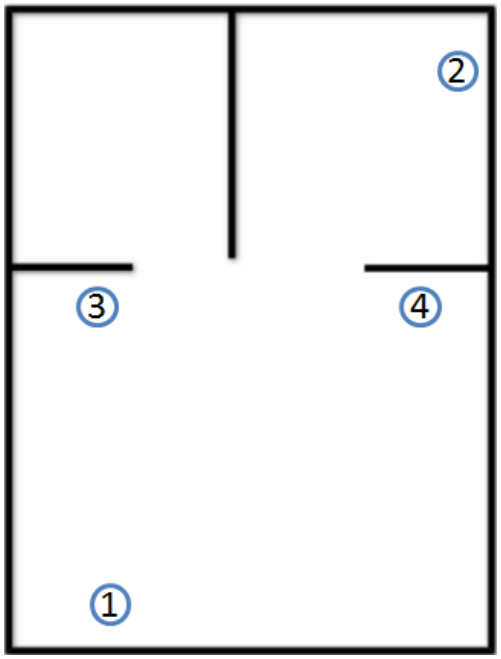


Figure 5.15: Locations 1-4 are areas in the environment in which using the proposed informative prior yields visibly better results.

grid square of side 2 for 3 sensor readings per robot pose, Figure 5.21(a) and for 1 sensor reading per pose, Figure 5.21(b). In this case the F_2 for the map using the proposed informative prior starts at a very high value because for these results we use a perfectly extracted prior. Therefore as sensor readings are obtained and sensor reading errors are incorporated the F_2 value drops. Even in such a scenario, however, the F_2 value obtained using the informative prior is consistently higher than that obtained using the non-informative prior. These results also demonstrate how using an accurate prior could help overcome problems caused by sensor reading errors. If the sensor readings greatly deviate from the prior the prior value could be favoured over the sensor reading to produce a more accurate map.

It is also worth noting that in both Figure 5.21(a) and (b) using the informative prior yields a final F_2 metric around 0.4 whereas in the case of a quick exploration, Figure 5.21(b), the non-informative prior only manages to achieve an F_2 of only 0.2. Therefore if a quick exploration is required for a safety critical mission using the proposed informative prior can be very beneficial. .

5.4 Concluding remarks

In this chapter the benefits of using an informative prior were explored, both in a qualitative and a quantitative sense, highlighting the benefits of using an

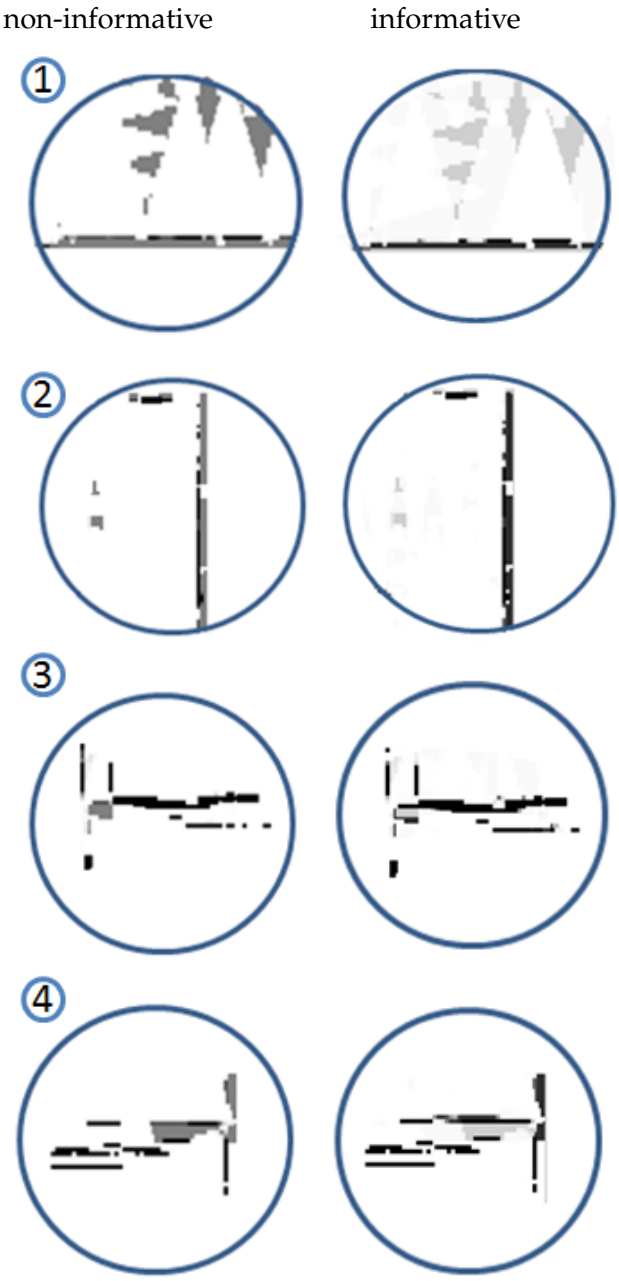


Figure 5.16: Zoomed-in areas in the final map produced using a grid square of side 2 and using 3 sensor readings per robot pose at locations 1-4, Figure 5.15. The left column shows sections of the map produced using the non-informative prior and the right column maps produced using the proposed informative prior. Grey areas in the left column correspond to a probability of occupancy of 0.5, providing no information about the environment.

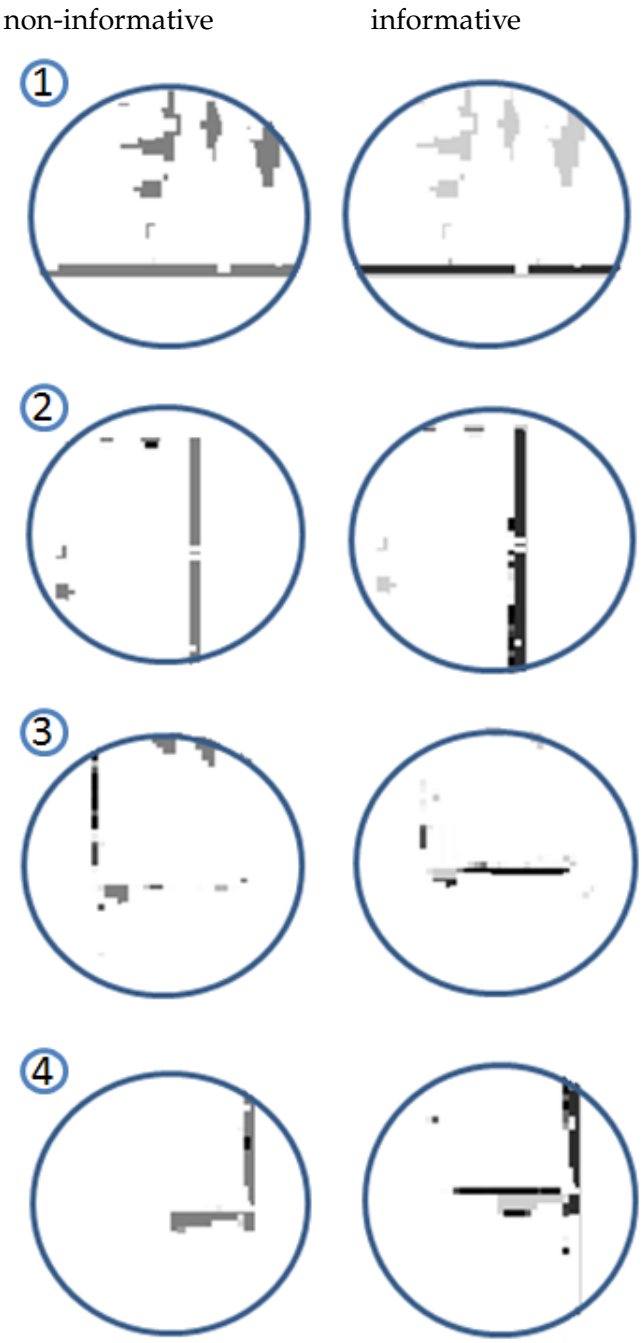


Figure 5.17: Zoomed-in areas in the final map produced using a grid square of side 3 and using 3 sensor readings per robot pose at locations 1-4, Figure 5.15. The left column shows sections of the map produced using the non-informative prior and the right column maps produced using the proposed informative prior. Grey areas in the left column correspond to a probability of occupancy of 0.5, providing no information about the environment.

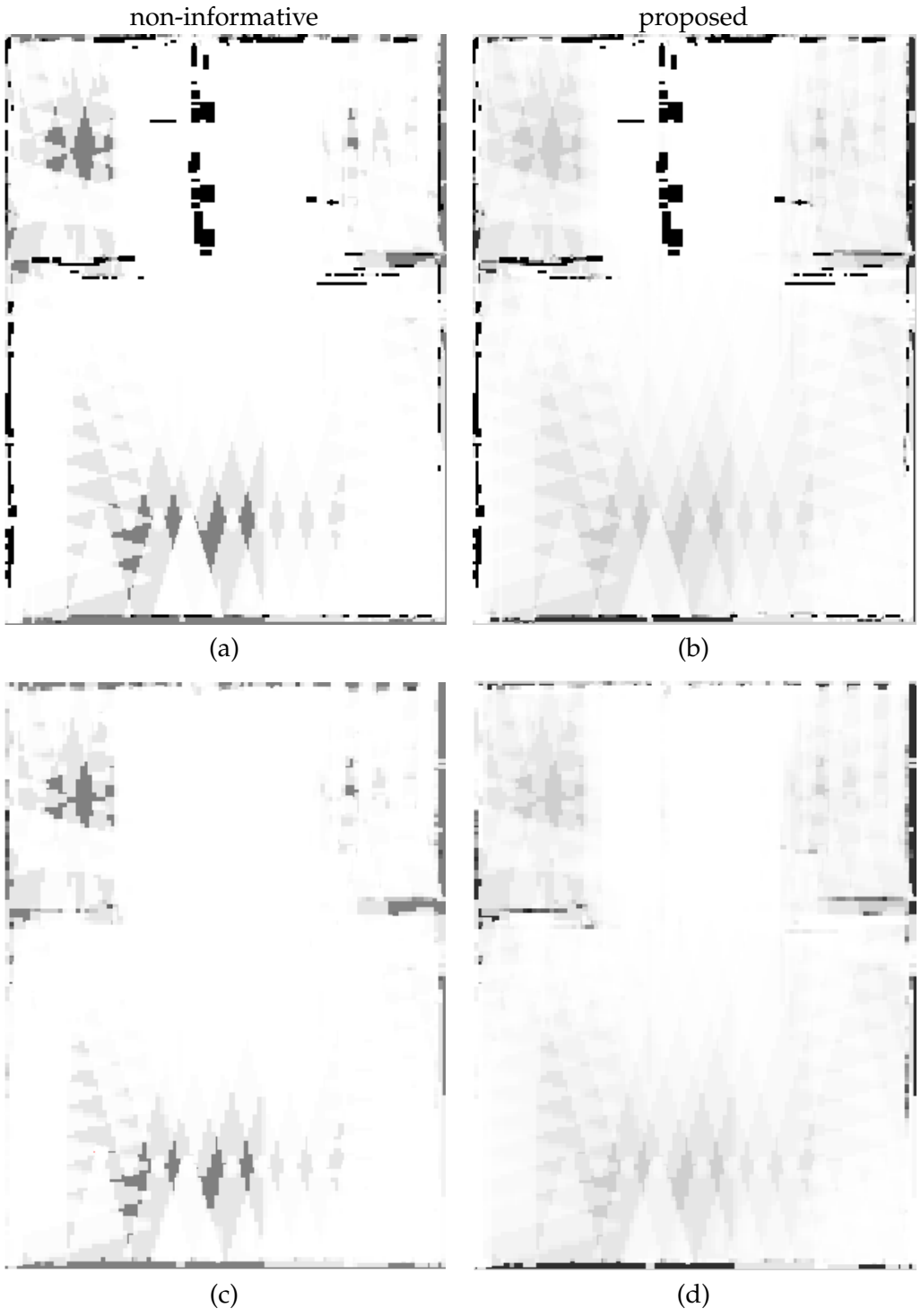


Figure 5.18: Final maps produced using the non-informative prior, (a), (c) and maps produced using the proposed informative prior, (b), (d), using 1 sensor reading per robot pose. Maps (a) and (b) were produced using a finer grid, with a grid cell side of 2, and (c) and (d) were produced using a coarser grid, with a grid cell side of 3. In maps (a) and (c) grey areas such as the top right and bottom left in (c) correspond to a probability of 0.5, providing no information about the environment.

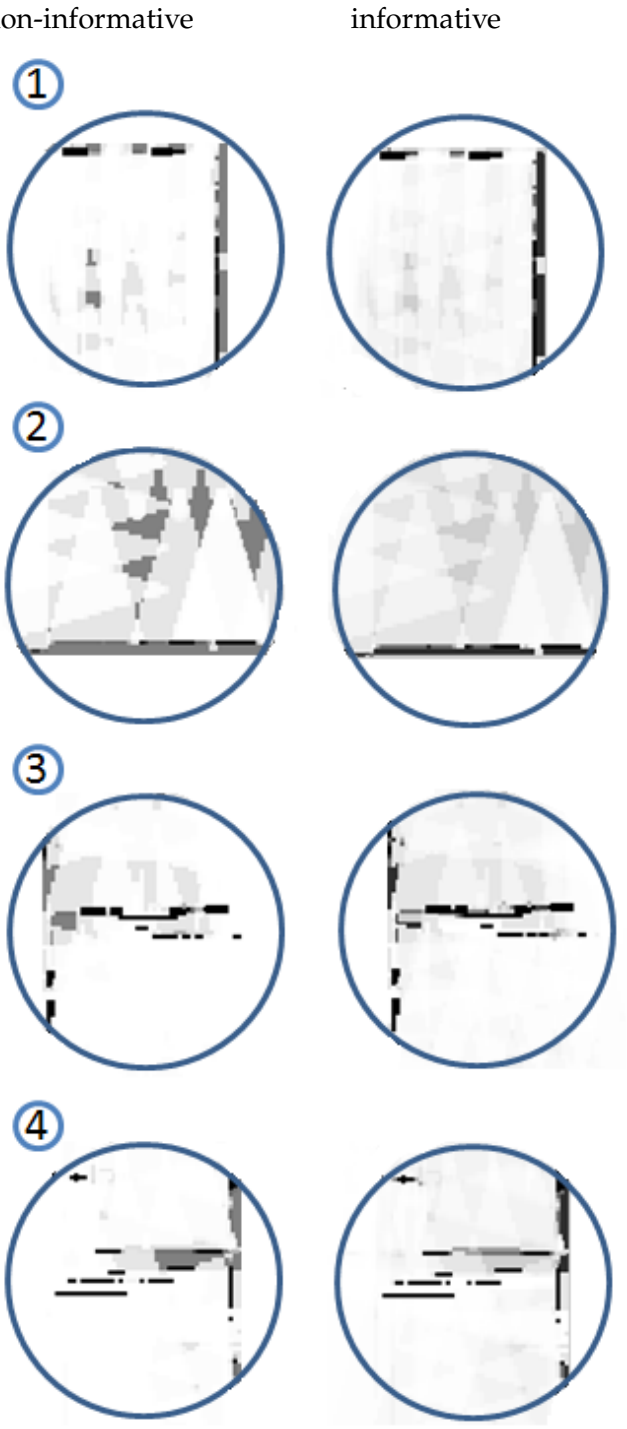


Figure 5.19: Zoomed-in areas in the final map produced using a grid square of side 2 and using 1 sensor reading per robot pose at locations 1-4, Figure 5.15. The left column shows sections of the map produced using the non-informative prior and the right column maps produced using the proposed informative prior. Grey areas in the left column correspond to a probability of occupancy of 0.5, providing no information about the environment.

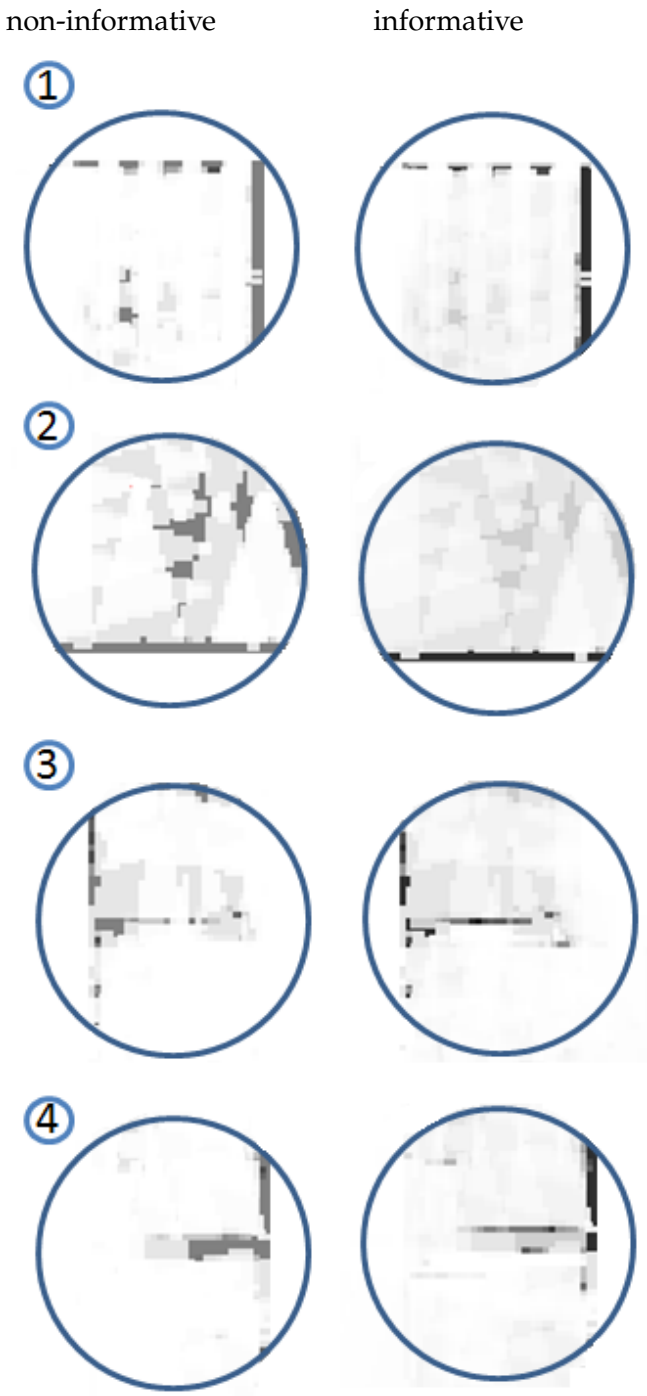
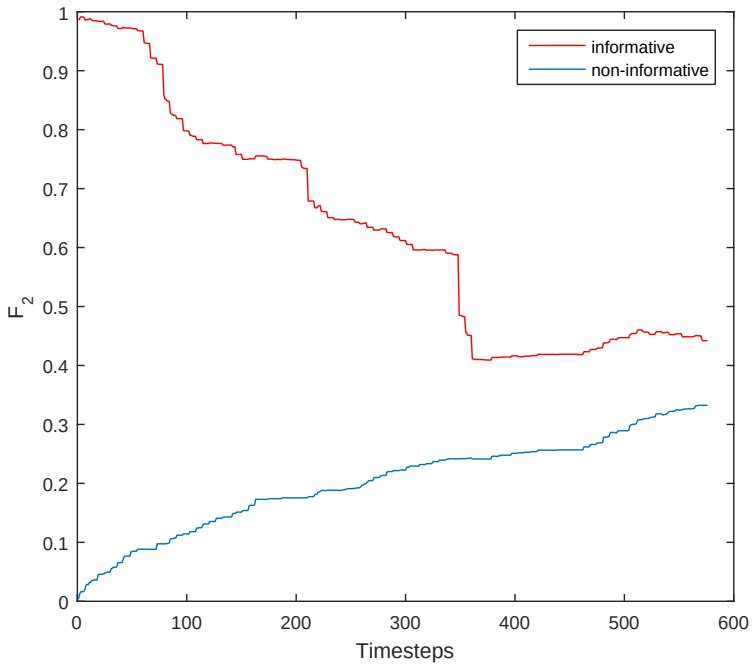
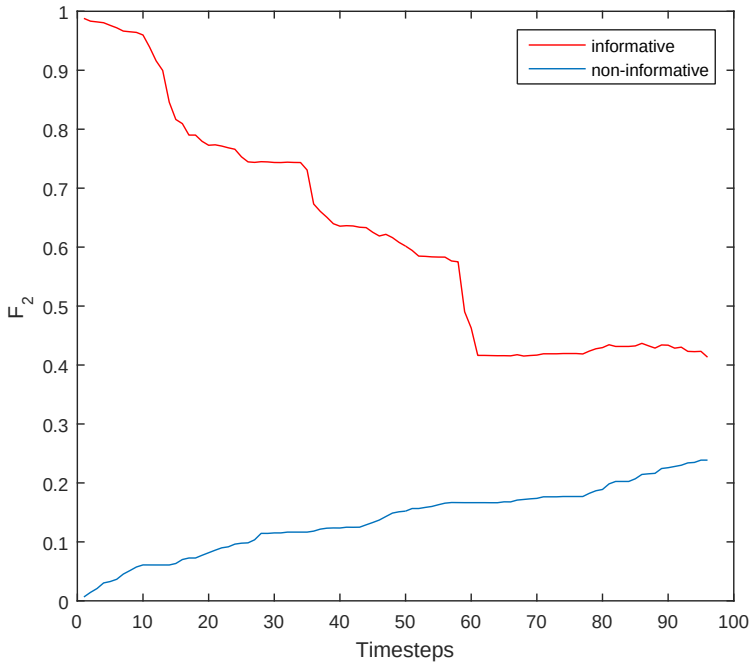


Figure 5.20: Zoomed-in areas in the final map produced using a grid square of side 3 and using 1 sensor reading per robot pose at locations 1-4, Figure 5.15. The left column shows sections of the map produced using the non-informative prior and the right column maps produced using the proposed informative prior. Grey areas in the left column correspond to a probability of occupancy of 0.5, providing no information about the environment.



(a)



(b)

Figure 5.21: A comparison of the F_2 metric at each time step as the environment is explored using an informative and a non-informative prior, using data collected with the Kinect sensor for a grid square of side 2. (a) 3 sensor readings per pose and (b) 1 sensor reading per pose. The map produced using an informative prior starts at a very high F_2 value because we assume a perfectly extracted prior and maintains a higher F_2 metric throughout.

informative prior. The proposed informative prior was found to yield an increase in the F_2 metric of over 20% compared to the non-informative prior. The use of informative priors was also found to have a number of other benefits. It provides information about unexplored areas if a fast exploration of the environment is required. It also provides information about the environment in case of sensor malfunction, errors or failure, producing more robust solutions for challenging environments such as USAR.

Chapter 6

Using optimised priors in SLAM

The previous chapter investigated how constructing and using optimised occupancy grid priors can lead to an improvement in occupancy grid map quality. So far we have assumed perfect knowledge of the robot's poses, testing only the effects of the priors on map quality. This chapter removes this assumption and explores how such priors can be integrated to existing SLAM solutions that estimate poses based on noisy odometry data. Experimental results obtained using the proposed informative prior are presented and compared to those obtained using a non-informative prior, confirming the improved performance when using an informative prior. The results presented in this chapter are published in the International Journal of Robotics Research [60].

The contributions of this chapter are:

- Experimental results of using SLAM with the proposed prior in a real indoors environment
- A comparison of results obtained using SLAM with an informative and a non informative prior
- A comparison of localisation accuracy when using the proposed extracted walls to construct a prior and using the drawing as-is

This chapter is organised as follows. Section 6.1 presents the algorithms implemented to perform SLAM, including the algorithms used for localisation, the motion and measurement models and the map update and inverse sensor model. Section 6.2 presents the software implementation of algorithms detailed in Section 6.1, relevant parameter values used and the experimental setup used to obtain results. Section 6.3 presents experimental results, comparing performance when using informative and non-informative priors, confirming that using the proposed

Algorithm 6.1 Occupancy grid FastSLAM implementation

for $k = 1 : T_{end}$ **do**
  $[\mathbf{x}_k^{estimated},\ Particles_k] = \text{Localise}(Particles_k, \mathbf{u}_k, \mathbf{z}_k)$
  $\mathbf{m}_k = \text{UpdateOccupancyGridMap}(\mathbf{z}_k, \mathbf{x}_k^{estimated}, \mathbf{m}_{k-1})$
end for

Algorithm 6.2 Localise$(Particles_k, \mathbf{u}_k, \mathbf{z}_k)$

$\overline{Particles_t} = Particles_t = \emptyset$
for j=1:M **do**
  $\mathbf{x}_k^{[j]} = \text{SampleMotionModel}(\mathbf{u}_k, \mathbf{x}_{k-1}^{[j]})$
  $\mathbf{w}_k^{[j]} = \text{MeasurementModel}(\mathbf{z}_k, \mathbf{x}_k^{[j]}, \mathbf{m}_{k-1}^{j})$
  $\overline{Particles_k} = \overline{Particles_k} + < \mathbf{x}_k^{[j]}, \mathbf{w}_k^{[j]} >$
end for
for j=1:M **do**
  draw with probability $\propto \mathbf{w}_k^{[j]}$
  add $< \mathbf{x}_k^{[j]} >$ to $Particles_k$
end for
$\mathbf{x}_k^{estimated}$ = mean(highest weighted particles)
return $\mathbf{x}_k^{estimated},\ Particles_k$

prior yields improved results. It also compares the performance of localisation when using the extracted walls to construct a prior and when using a map as-is, with the extracted walls yielding significantly more accurate localisation results.

6.1 SLAM implementation

An occupancy grid FastSLAM implementation was used to obtain experimental results, as shown in Algorithm 6.1. At each time step, for each particle, the motion model is sampled and the measurement model is used to update particle weights. Particles are then re-sampled and the mean of the highest weighted particles is used to update the map. The map is updated using the occupancy grid mapping algorithm.

Localisation is performed using Monte Carlo Localisation as described in [134] and shown in Algorithm 6.2.

Motion model

This model assumes that the robot uses pure rotation and translation motions to get from one location to another. This is shown in Algorithm 6.3 [134], where the pose is $\mathbf{x}_{k-1} = (x\ y\ \theta)$ and the control $\mathbf{u}_k = (\hat{\mathbf{x}}_{k-1}\ \ \hat{\mathbf{x}}_k)$ is a differentiable set of two pose estimates obtained by robot's odometer. In an odometry model,

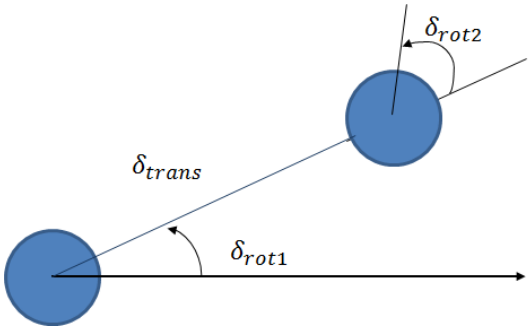


Figure 6.1: The odometry motion model assumes that the robot uses pure rotation and translation motions to get from one location to another. It approximates motion by a rotation, δ_{rot1}, then a translation, δ_{trans}, and a second rotation, δ_{rot2}.

Algorithm 6.3 SampleMotionmodel$(\mathbf{u}_k, \mathbf{x}_{k-1}^{[j]})$

$\delta_{rot1} = atan2(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$
$\delta_{trans} = \sqrt{(\bar{y}' - \bar{y})^2, (\bar{x}' - \bar{x})^2}$
$\hat{\delta}_{rot1} = \delta_{rot1} - \textbf{sample}(\alpha_1\delta_{rot1}^2 + \alpha_2\delta_{trans}^2)$
$\hat{\delta}_{trans} = \delta_{trans} - \textbf{sample}(\alpha_3\delta_{trans}^2 + \alpha_4\delta_{rot1}^2 + \alpha_4\delta_{rot2}^2)$
$\hat{\delta}_{rot2} = \delta_{rot2} - \textbf{sample}(\alpha_1\delta_{rot2}^2 + \alpha_2\delta_{trans}^2)$
$x' = x + \hat{\delta}_{trans}cos(\theta + \hat{\delta}_{rot1})$
$y' = y + \hat{\delta}_{trans}sin(\theta + \hat{\delta}_{rot1})$
$\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$
return $x_k = (x', y', \theta')$

motion is approximated by a rotation, δ_{rot1}, then a translation, δ_{trans}, and a second rotation, δ_{rot2}, Figure 6.1. Parameters α_1-α_4 are error parameters, corresponding to rotational error due to rotational motion (α_1), rotational error due to translational motion (α_2), translational error due to translation motion (α_3) and translational error due to rotational motion (α_4) respectively.

Measurement model

A likelihood sensor field model was used, shown in Algorithm 6.4 [134], with $\textbf{prob}(dist, \sigma_{hit})$ computing the probability of the distance under a zero mean Gaussian with standard deviation σ_{hit} [134]. The measurement model models three types of sensor noise:

- Measurement noise: modeled using a zero mean Gaussian, $\epsilon_{\sigma_{hit}}(dist)$, where $dist$ is the Euclidean distance between the measurement and the nearest obstacle in the map, Algorithm 6.4
- Failures: modeled using a point-mass distribution at z_{max}

Algorithm 6.4 MeasurementModel($\mathbf{z}_k, x_k^{[j]}, \mathbf{m}_{k-1}^j$)

$q = 1$
for all j **do**
  if $z_k^j \neq z_{max}$ **then**
    $x_{\mathbf{z}_k}^j = x + x_{j,sens}cos\theta - y_{j,sens}sin\theta + z_k^j cos(\theta + \theta_{j,sens})$
    $y_{\mathbf{z}_k}^j = y + y_{j,sens}cos\theta + x_{j,sens}sin\theta + z_k^j cos(\theta + \theta_{j,sens})$
    $dist = \min x', y' \{\sqrt{(x_{z_k}^j - x')^2 + (y_{z_k}^j - y')^2} | (x', y') \; occupied \; in \; m\}$
    $q = q \times (z_{hit} \times \mathbf{prob}(dist, \sigma_{hit}) + \frac{z_{random}}{z_{max}})$
  end if
end for
return q

- Unexplained random measurements: modeled by a uniform distribution

Update occupancy grid map

The map was updated using the occupancy grid mapping algorithm, shown in Algorithm 6.5, [134]. The inverse sensor model used was given by

$$InverseSensorModel(\mathbf{m}_i, \mathbf{x}_k, \mathbf{z}_k) = \begin{cases} p_{free}, & points < z \\ p_{occupied}, & z < points < point_{max} \\ p_{previous}, & points > point_{max} \end{cases} \tag{6.1}$$

where $points$ correspond to pixels in the map, z is the distance from the robot at which an obstacle was found, $point_{max}$ is the furthest away point of a cell containing the obstacle, p_{free} the value assigned by the model to free cells and $p_{occupied}$ the value assigned by the model to occupied cells, Figure 6.2.

6.2 Experimental setup

This section details the software used to implement the algorithms detailed in the previous section as well as the hardware used and the environment explored.

6.2.1 Robot and environment

In this chapter the use of informative priors was evaluated in a large scale, realistic experiment using a standard SLAM algorithm, FastSLAM for occupancy grids, as

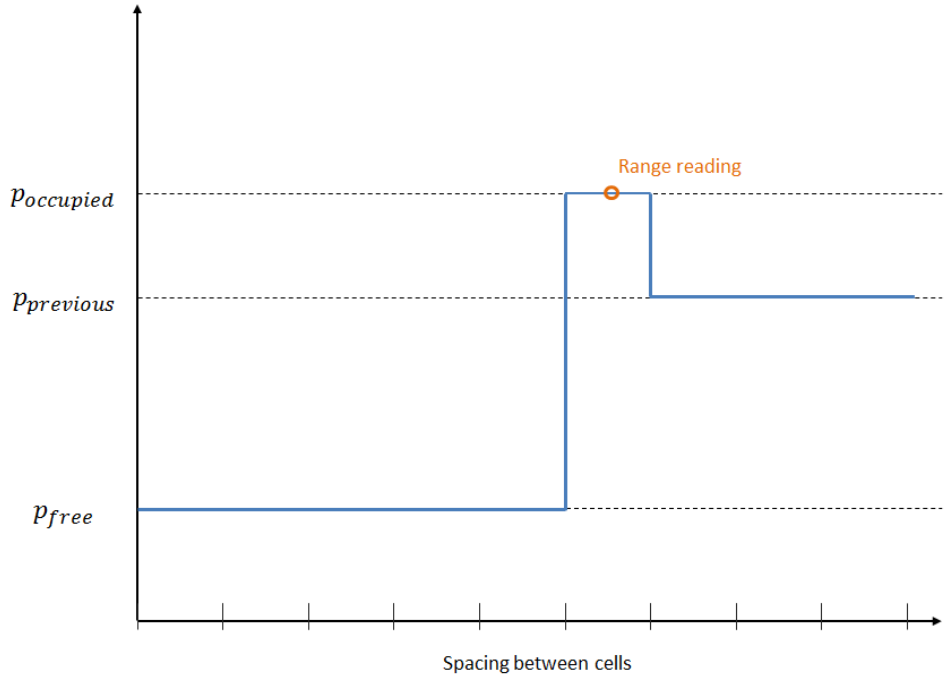


Figure 6.2: The inverse sensor model used assigns a probability p_{free} to cells closer than the detected obstacle, $p_{previous}$ to those beyond the cell in which the obstacle is detected and $p_{occupied}$ to the cell in which the obstacle lies.

Algorithm 6.5 UpdateOccupancyGridMap$(\mathbf{z}_k, \mathbf{x}_k^{estimated}, \mathbf{m}_{k-1})$

for all cells m_i **do**
  if m_i in perceptual field of $\mathbf{z}_t$ **then** $l_{k,i} = InverseSensorModel(\mathbf{m}_i, \mathbf{x}_k, \mathbf{z}_k) + l_{k-1,i} - l_{0,i}$
  else $l_{k,i} = l_{k-1,i}$
  end if
end for

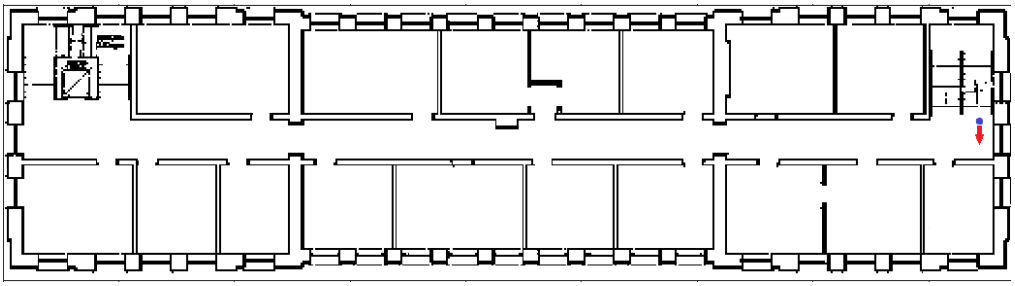


Figure 6.3: Initial turtlebot pose in large scale experiment: the blue dot indicates its (x, y) position and the red arrow its heading, θ.

detailed in Algorithm 6.1. The experiment consisted of localising and mapping through one floor of a University of Sheffield Engineering building consisting of offices, with an approximate size of 10×40 metres ($400m^2$). The experiment was realistic in the sense that rooms were cluttered with usual everyday material, and were not emptied or altered to simplify the environment.

A turtlebot with an onboard Kinect sensor was used, as described in Chapter 5. A floor plan of the building was obtained before exploration and walls were extracted, obtaining a nearly perfectly extracted set of walls, Figure 6.4. The scale of the drawing was used to set the resolution of the occupancy grid prior and to determine the robot starting pose in map coordinates. A suitable starting pose was chosen, placing the robot near stairs which a rescuer or operator would be able to access. All the rescuer/operator would then need to do would be to upload the map, set the correct starting pose and set the robot there, then explore the environment using tele-operation.

The turtlebot was placed at the location shown in Figure 6.3 facing the first office on the bottom right corner of the floor plan. A tape measure was used to determine the distance of the turtlebot from the stairs and the wall to the right and the turtlebot heading was set ensuring the Kinect was parallel to the stairs. Using the drawing scale the measured (x, y) starting pose was converted to map coordinates.

6.2.2 fastSLAM algorithm software implementation

The fastSLAM algorithm detailed in Algorithm 6.1 was implemented using the parameters shown in Table 6.1. The Matlab 2016b robotics toolbox was used for this SLAM implementation [1]. The robotics toolbox provides a sensor and motion model tailored to a turtlebot mounted with a Kinect sensor, making it a suitable choice for development.

The toolbox was modified in three ways:

Parameter	Value
Resampling interval	1
Particle limits	[1500 4500]
$\alpha_i,\ i = 1...4$	0.2
Measurement limits (m)	[0 12]
z_{max}	2
Number of beams (ray model)	60
σ_{hit}	0.2
z_{random} weight	0.05
p_{free}	0.8
$p_{occupied}$	0.2
p_{wall}	0.27
p_{space}	0.72

Table 6.1: Model parameters used in SLAM implementation.

- The map was updated using a modified version of the inbuilt `insertRay` method - see Updating the map section below
- The AMCL class `robotics.MonteCarloLocalization` method was modified- the map was converted to a tunable property, allowing the use of the updated map to localise at each time step
- The visual quality of the final map was improved - see Map visualisation section below

An initial pose with zero covariance was provided, since the robot's starting pose was known.

Updating the map

Following the theoretical analysis of Section 2.3.4 and the multi-objective optimisation performed using experimental data, prior values of $p_{wall} = 0.27$ and $p_{space} = 0.72$ were assigned using this sensor model. This slightly lower p_{wall} value was found to perform better using this more sophisticated sensor model. The Matlab toolbox leverages the fact that $l(0.5) = 0$ to remove the term $-l_{0,i}$ from the calculation in

$$l_{k,i} = InverseSensorModel(\mathbf{m}_i, \mathbf{x}_k, \mathbf{z}_k) + l_{k-1,i} - l_{0,i} \tag{6.2}$$

The inbuilt method was modified to add back the term and allow an update using prior values.

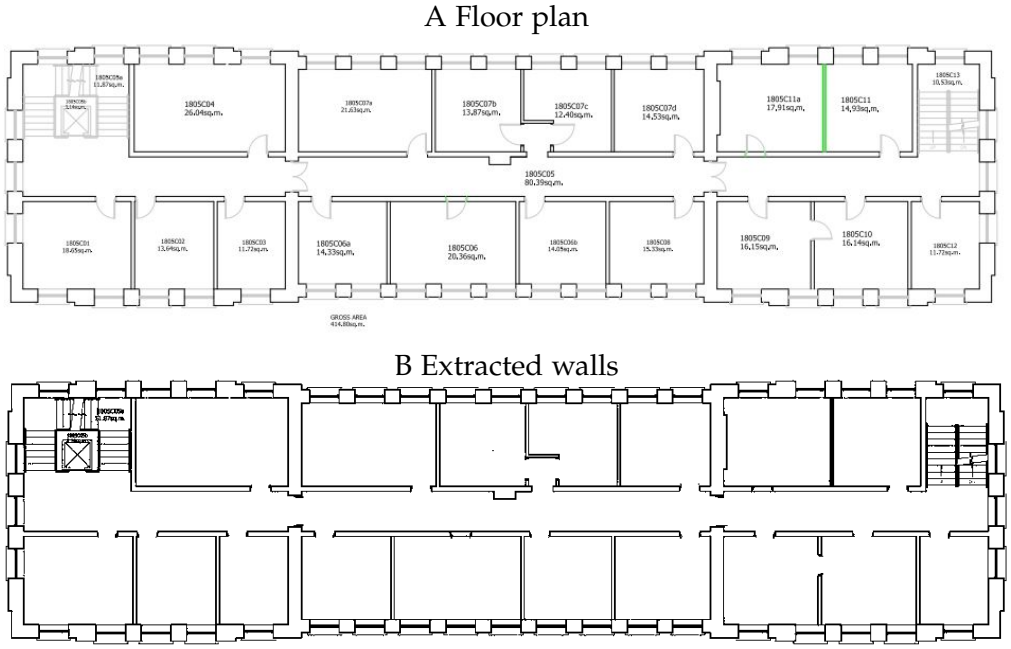


Figure 6.4: Large scale experimental results floor plan. A Building floor plan; B Extracted walls and empty space.

Map visualisation

When using this approach, the maps contained gaps along the depth of walls since occupied space is only marked where the obstacle is reached. Visually, this results in walls being represented as rectangles that have dark sides but are white in the middle. This visual quality issue was mitigated by increasing the number of black pixels assigned to detected obstacles to take into account obstacle width.

6.3 Experimental results and discussion

Data was collected for two experimental runs, a nearly complete exploration and a largely incomplete exploration, to highlight the benefits of using an informative prior in each case. The results obtained using the prior map were compared to those obtained using a non-informative prior. The floor plan was used for localisation for both the informative and the non-informative prior case to allow a direct comparison of the effects of different priors on mapping without taking into account the effects of the prior on localisation quality. Not using the floor plan to localise would lead to a less accurate map for the non-informative prior, which highlights a further advantage of using the prior map: more accurate localisation and the ability to start with a known, accurate $\mathbf{x}_0$.

6.3.1 SLAM using informative and non-informative priors

The walls used to construct the prior map in this case, Figure 6.6A, are not perfectly extracted, with some of the doors appearing as wall sections on the sixth room in the top row and the fifth room in the bottom row of offices. The final map is unaffected by these small inaccuracies which are corrected during mapping.

Experiments were also run using a less accurate $\mathbf{x}_0$, using particles initialised with a non-zero variance

$$Covariance = \begin{vmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{vmatrix} \tag{6.3}$$

The resulting maps were less accurate but still better than those produced using a non-informative prior. For lower variance values the results were very similar to those produced using a perfectly accurate $\mathbf{x}_0$. Assuming a known prior pose is realistic since, given the extracted prior and a floor plan, an operator can align the robot to a known location before starting exploration.

The results of these experiments are shown in Figure 6.6 following the exploration path shown in Figure 6.5. Using the proposed prior yields a more accurate map both in a qualitative and quantitative sense. The map produced using the proposed prior yields an F_2 value that is over 50% higher than that achieved using the non-informative prior. It also yields more correctly mapped walls, demonstrating the advantages of using the proposed prior over a non-informative one. These results demonstrate an added benefit of using a prior map. If exploration of the whole floor were not possible due to, for example, limited exploration time in a time-critical mission or certain areas being inaccessible, the proposed prior provides some information about unexplored areas. This may not be completely accurate but it would allow a human operator or rescuer to better interpret the mapped sections. The map shown in Figure 6.6B is less easy to interpret and it is less clear what sections of the building the map areas correspond to.

In all cases there appears to be some noise in the map, particularly within the detected rooms. That is due to the fact that the environment explored was a real world, cluttered office building, containing furniture. Therefore objects detected within the rooms mostly correspond to desks, chairs, bookcases and half-open doors. Since our aim is to improve performance in real world environments we did not aim to simplify the environment by only exploring long corridors (as is often done in the literature) or emptying the rooms.

These results further confirm that using the proposed informative prior can improve performance especially when a quick exploration is required, yielding

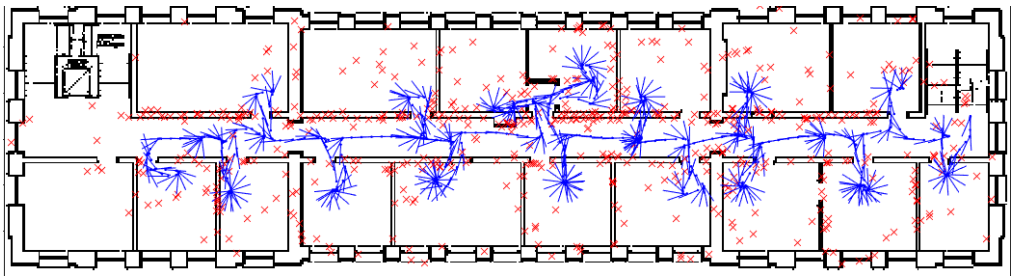


Figure 6.5: Large scale complete exploration path followed produced by SLAM: robot path in blue, edges of detected occupied segments shown in red crosses.

consistently better quantitative and qualitative results.

Incomplete exploration

Experiments were also run for an incomplete exploration of the environment as shown in Figure 6.8, following the trajectory shown in Figure 6.7. These results show further benefits of using the proposed prior, with the map proposed using the non-informative prior giving little information about the building's structure.

In a scenario where some of the rooms are inaccessible, using the proposed prior provides information that makes it possible to interpret the map produced by the robot and gives an idea of the structure of the building.

6.3.2 Localisation quality

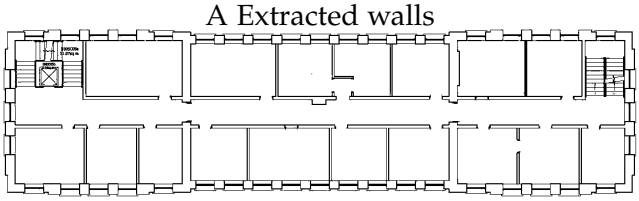
Pure AMCL was run using sensor and odometry data collected for the full exploration and the floor plan as-is to localise. The results were compared to those obtained using the extracted walls to localise, Figure 6.9. When using the plan as-is the quality of the localisation is significantly reduced, producing an inaccurate exploration path towards the end of exploration.

These results indicate that if the map is used to localise large lines in the drawing corresponding to suggested furniture, labeling or dimensions could cause localisation errors. Even in this case, where discrepancies between the drawing as is and the extracted prior are relatively small the quality of localisation drops over time when using the drawings as-is. This further confirms that the floor plan needs to be processed to extract walls before using it to construct a prior map.

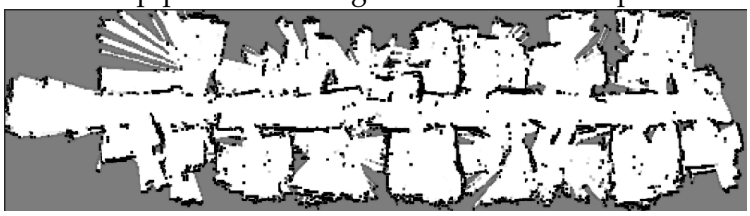
6.4 State-of-the-art occupancy grid SLAM

State-of-the-art occupancy grid SLAM implementations do not incorporate prior information to update an occupancy map. DP SLAM uses a sensor model and sen-

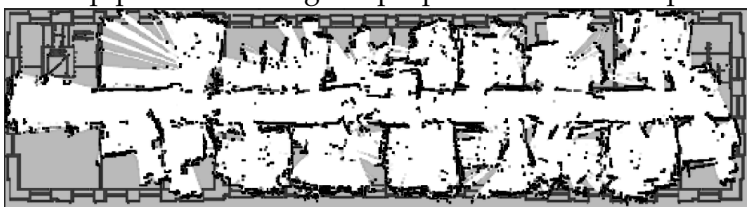
A Extracted walls



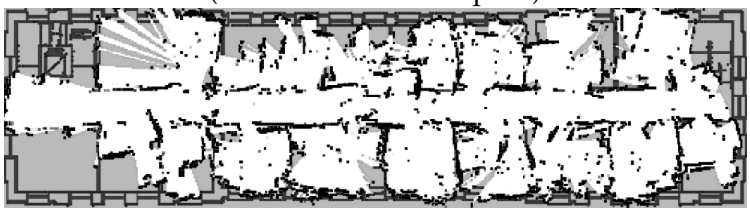
B Map produced using a non-informative prior



C Map produced using the proposed informative prior



D Map produced using the proposed informative prior; $\mathbf{x}_0$ with a variance of 0.1 (less accurate initial pose)



E Evolution of F_2 with time

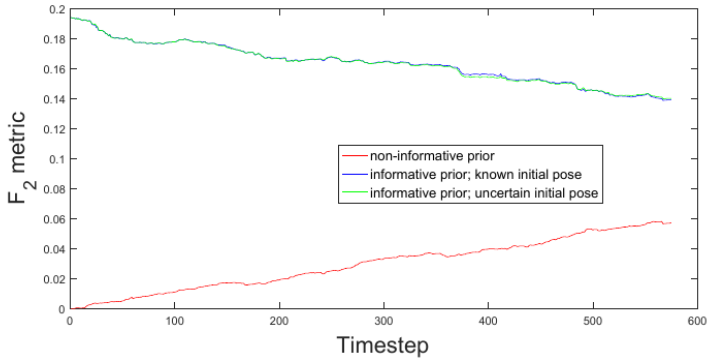


Figure 6.6: Large scale experimental results. A Extracted walls and empty space; B Map produced using a non-informative prior; C Map produced using the proposed prior, successfully mapping more walls than using the non-informative prior; D Map produced using the proposed informative prior and a slightly less accurate $\mathbf{x}_0$; E Evolution of the F_2 metric with time for an informative and non-informative prior.

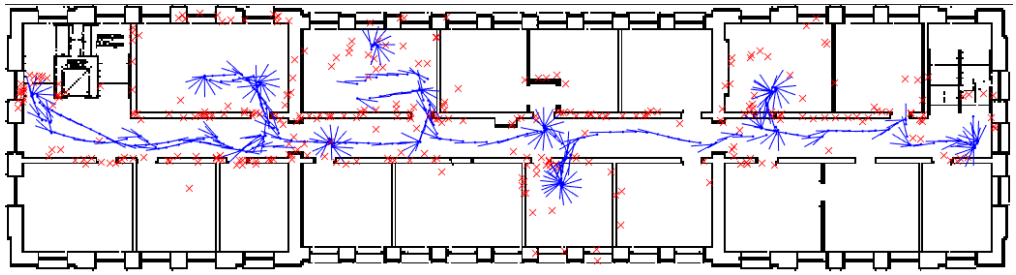


Figure 6.7: Large scale complete exploration path followed produced by SLAM for an incomplete exploration: robot path in blue, edges of detected occupied segments shown in red.

sor data to update a value of occupancy, with unexplored areas being assigned an unknown value and the prior playing no role in the update. This approach facilitates updating the tree structure that maintains the values of the occupancy grid. The ROS module for gmapping, arguably one of the most commonly used SLAM implementations that update an occupancy map, also uses this representation of occupancy.

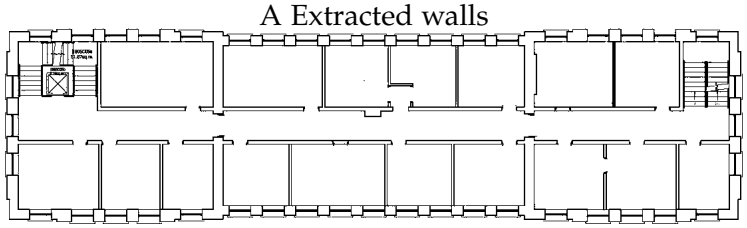
These implementations assume that modern sensors such as laser range scanners are accurate and thus sensor readings can be used by recursively updating values of occupancy using only sensor data. This approach has a number of drawbacks, however:

- More accurate sensors tend to be more expensive
- Sensor errors due to reflective surfaces or moving objects cannot be corrected easily
- Sensor failures or errors due to dust interfering with the sensor, for example, cannot be corrected
- In case of sensor malfunction the robot has no information about its environment

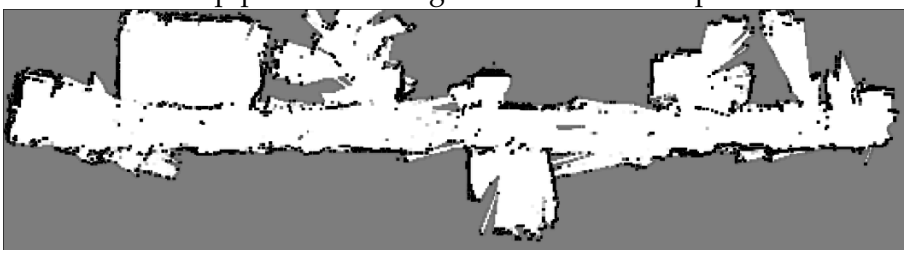
Incorporating the proposed prior achieves computational savings as well as map quality improvements. Only one map is maintained, updated using the best pose estimate at each time step. Since the initial pose is known and a prior map of the environment is available to help localise more accurately, this approach yields good results. Using the proposed prior has the added benefit of allowing a visualisation of the building before and after mapping, using a recursive map update which yields more accurate mapping results.

This analysis highlights the benefits of using a prior but also indicates a need for more flexible systems that incorporate both prior information and sensor infor-

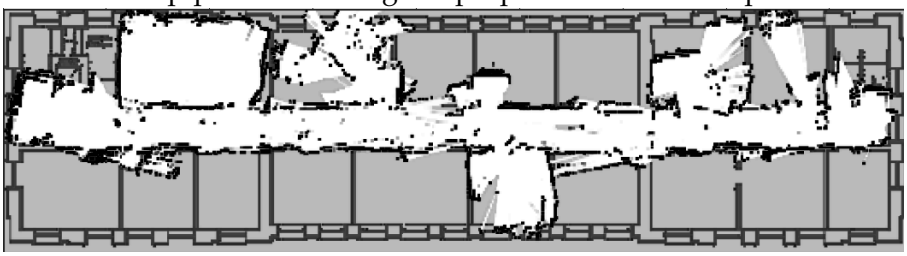
A Extracted walls



B Map produced using a non-informative prior



C Map produced using the proposed informative prior



D Evolution of F_2 with time

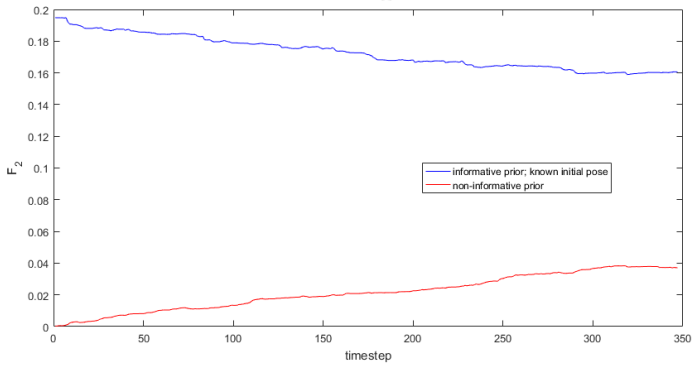
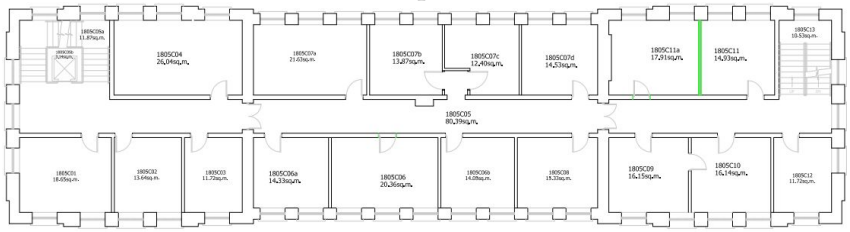
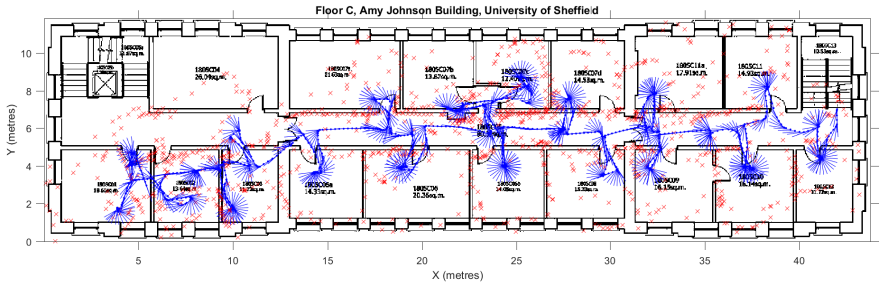


Figure 6.8: Large scale experimental results for an incomplete exploration. A Extracted walls and empty space; B Map produced using a non-informative prior; C Map produced using the proposed prior, successfully mapping more walls than using the non-informative prior; D Evolution of the F_2 metric with time for an informative and non-informative prior.

A Floor plan as-is



B Path produced by AMCL using floor plan as-is



C Path produced by AMCL using the extracted walls

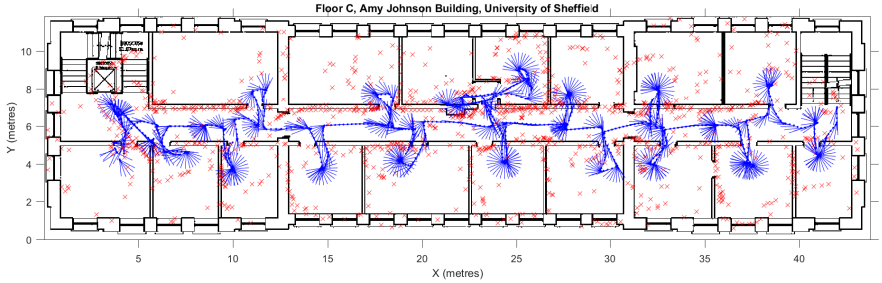


Figure 6.9: Large scale experimental results using the floor plan as-is for the prior map. A Floor plan as-s used with AMCL to produce the trajectory in B; B Robot path produced by AMCL in blue, edges of detected occupied segments in red using map as-is; C Robot path produced by AMCL in blue, edges of detected occupied segments in red using extracted walls.

mation, allowing them to rely on the source of information that is more accurate to correct erroneous map updates

6.5 Concluding remarks

This chapter presented the SLAM implementation used, which incorporates an informative prior map constructed using the analysis presented in this thesis. Real world experiments were conducted using a turtlebot exploring a floor of an office building and results obtained using the informative and non-informative priors were presented. The maps produced using the proposed method were found to outperform maps produced using a non-informative prior both in terms of qualitative and quantitative performance. The importance of using extracted walls to construct a prior was also highlighted, since it affects localisation quality.

Chapter 7

Conclusions and future work

7.1 Summary and conclusions

The vast number of available SLAM solutions require a number of simplifications or conditions to hold true in order to perform well in real life applications. For example, Kalman filter-based methods require linearisations of complex non-linear robot dynamics and DP SLAM assumes accurate sensor readings. Such simplifications are not realistic, however, leading to poor performance in real life applications. A need to produce SLAM solutions that perform well even in challenging scenarios, with limited computational resources or when a fast exploration is required in time-critical applications was thus identified, prioritising realistic over mathematically elegant solutions.

In order to address this need, this thesis has explored the idea of using informative Bayesian priors to improve the performance of SLAM algorithms without increasing the computational complexity of the SLAM algorithm. It presented indoors occupancy grid FastSLAM as a case study, with architectural drawings and floor plans used as a source of prior information. The main aim was to propose a method to construct an optimised Bayesian indoors prior map and demonstrate how using such a prior can improve SLAM performance.

This thesis has proposed a framework for the construction of informative Bayesian prior maps in SLAM to improve map quality, proposing a complete method to produce optimised informative indoors mapping priors using architectural drawings and floor plans. The aim of this research is to help produce SLAM solutions that perform better even when limited computational resources are available, a quick exploration is required and/or unreliable or low cost sensors are used. Using an informative prior only incurs a one-off cost to produce the prior map but does not add to the computational cost of running the SLAM algo-

rithm itself and can be used with any SLAM implementation that uses a recursive occupancy grid mapping algorithm to update the map.

Prior relevant information had to be extracted from architectural drawings and floor plans to construct such a prior. Whilst there are a number of methods commonly used to extract information from architectural drawings the application of such research is drawing archiving or conversion of 2D drawings to 3D representations. Given this focus none of these methods prioritise wall detection and as such are ill-suited to extract useful information for prior construction.

This gap in the literature was addressed by proposing a novel method to detect doors and walls in architectural drawings and floor plans. A set of geometric constraints and tests were performed to detect doors in the drawings by approximating doors by isosceles triangles. Walls were then detected leveraging the fact that all doors are attached to walls. Drawing processing algorithms are commonly assessed using methods such as calculating the percentage of features identified correctly. This thesis formulated this assessment as a binary classification problem, testing whether each pixel in the input image was classified correctly as a wall or empty space. Precision, recall and false positive rate were then used to assess performance for representative drawings, yielding precision of at least 98%, recall of at least 61% and a false positive rate lower than 0.09 % for all drawings. These results indicate that the proposed method may not always detect all walls in the image as indicated by the lower recall values but it very rarely detects walls incorrectly, as indicated by the very high precision and low false positive rate.

A method to convert the walls extracted from drawings into Bayesian priors was then devised. Occupancy grids were used as a map representation and an occupancy grid mapping simulator was produced in Matlab to test the effects of assigning different priors of occupancy to detected walls and empty space. A new approach to assessing map quality was proposed, formulating the problem as a classification of all map cells as being occupied or empty. This approach allowed the use of common classification metrics, precision and recall, to analyse performance.

The objectives of maximising recall and maximising precision were found to be conflicting. A multi-objective optimisation genetic algorithm was thus used to identify Pareto optimal prior values $(p_{wall},\ p_{space})$ to assign to detected walls and empty space in the architectural drawing. This optimisation was performed for representative drawings, including both floor plans and architectural drawings. A number of Pareto optimal values common to all drawings were found to exist and a further study was conducted to find $(p_{wall},\ p_{space})$ values that yield good qualitative performance for all drawings.

The Pareto Optimal $(p_{wall},\ p_{space})$ pairs for all drawings were processed as a single dataset. A clustering algorithm was then used to identify clusters in the data and facilitate the search for Pareto optimal prior values $(p_{wall},\ p_{space})$ that yield good qualitative performance for all drawings. Three clusters were detected and representative maps were tested for each cluster to identify the cluster with values that yield good qualitative peformance. Once the cluster of solutions yielding the best qualitative maps was identified values within that cluster were tested and the values $p_{wall} = 0.2,\ p_{space} = 0.9$ were found to yield the best maps qualitatively. Therefore the values $p_{wall} = 0.2,\ p_{space} = 0.9$ were proposed as informative prior values to assign to detected walls and empty space in the prior map.

The potential benefits of using a prior map were then explored by benchmarking maps produced using the proposed informative prior against the maps produced using an uninformative prior. The use of the proposed informative prior was found to yield an increase in the F_2 metric of over 20%. In terms of qualitative performance the use of a prior map was found to yield improved performance in the majority of cases, even when a coarser grid resolution was used. Finally, the use of the proposed informative prior was found to yield improved performance even if a quick exploration was performed or an incomplete or partially inaccurate prior map was used.

The results obtained were confirmed experimentally using a turtlebot running occupancy grid fastSLAM that incorporates the proposed priors. These results confirm the results obtained in simulation and those obtained using a known pose.

7.2 Future work

There are a number of possible extensions to the work presented in this thesis as outlined in the following sections. Each section presents possible extensions to different aspects of the research presented in each chapter.

Drawing processing

The drawing processing method proposed in Chapter 3 is a framework that allows a conversion of information from a human-friendly format to a robot-friendly format. In this thesis the main focus has been accurate wall detection, focusing on detecting horizontal and vertical wall segments in the image. This framework can then be extended to detect more complex wall configurations such as walls at an angle or round buildings. Currently doors are detected but only to enable wall detection. The current door detection algorithm can be improved by using a large

dataset of possible door symbols to adjust existing door detection tests. Machine learning techniques can be used to determine more accurate triangle angle ranges that are accepted as true doors. This endeavour would require a large number of diverse drawings to be used as a training set and could be facilitated by gaining access to a database of digitised drawings.

Going beyond the existing algorithm, the features detected in the drawing can be extended to include stair detection to identify exit routes. It can also be updated to include suggested furniture detection which can be used to assign detected furniture locations a higher prior value than detected walls but lower than detected empty space. The drawing quality assessment method used can then be extended to a classification of pixels in the image as belonging to more than two possible classes. The prior value to assign to these could then be determined using the optimisation method described in Chapter 4. Windows can be detected to identify areas where laser range scanner readings may be unreliable or incorrect. Labelling text and dimension lines can be used to extract information about the map scale and thus facilitate the scaling between grid cells and real world dimensions.

Beyond SLAM priors Architectural drawings and floor plans contain a wealth of information that can be used to improve indoors robot operation. This thesis has explored the potential benefits of using extracted information to construct SLAM priors. This information can also be used to improve the integration of robots operating in indoors environments, helping robots localise more accurately. Incorporating more information such as door and stair locations can help robots integrate more smoothly into existing infrastructure, efficiently planning routes to access given locations. Further processing of labeled drawings to determine the location of certain rooms such as the location of a kitchen or living room, for example, could allow a human user to ask a robot to access a given room. This could be a useful feature for robots operating at homes such as robotic vacuum cleaners or robots providing medical supplies in hospitals.

Determining optimised prior values

A more extensive study of different prior values to assign to walls and empty space can be conducted using a larger set of drawings. Moreover, if features such as suggested furniture are detected a study can be conducted to determine an optimised prior value to assign to them. Intuitively such areas should be assigned a lower probability of being empty than detected empty space but a higher probability of being empty than detected walls.

The (p_{wall}, p_{space}) values examined in this thesis are not continuous and linear

interpolation is used to produce continuous values of precision and recall. A higher order interpolation method could be used or more (p_{wall}, p_{space}) pairs could be collected. This is unlikely to greatly affect the final results but would be worth testing.

A k-means clustering method is currently used to detect clusters in the Pareto optimal solutions data for all drawings. A larger drawing dataset would provide more data points and could thus result in more clusters being identified. A possible trend of different optimal prior parameters for different types of drawings could then be explored.

Finally, performing the optimisation for different sensor models will highlight optimal prior value choices for different sensors. The theoretical background established in this thesis as well as the method outlined to perform this optimisation can be used to test different sensor models and determine optimised prior values for commonly used models.

3D priors

The work presented in this thesis has focused on 2D mapping and as such used 2D drawings as a source of prior information. 3D models of buildings are becoming more accessible as standards such as BIM become more common. Using such models as a source of prior information, this work can be extended to create 3D priors for point cloud maps. This extension is well suited to UAVs, which can be used to perform 3D mapping.

Benefits of using informative priors in USAR missions

In order to convert these ideas to a system that can be used reliably in USAR an extensive study of real buildings, drawings and sensor data would be required, carefully selecting case studies that explore a variety of drawing representations and building structures. Creating a ROS module that incorporates the proposed optimised priors to gmapping and uses recursive probabilistic mapping rather than tree representations would be a next step.

The benefits of using such priors could then be tested in scenarios where sensors behave unreliably or fail. This method could also be tested using low-cost sensors that are less accurate to study how using a prior can improve performance. The obtained results would then need to be benchmarked against currently used algorithms in challenging environments to quantify the benefits in terms of system reliability, computational efficiency, system cost and map consistency.

USAR specific extensions

In the USAR scenario the informative prior can also be used to produce maps that are more helpful to first responders. Assuming a reasonably accurate prior, building sections that differ greatly from their prior values can be highlighted as potentially collapsed areas, enriching the information provided by the map.

Sensors can often produce incorrect readings due to smoke or dust, malfunction or even fail in USAR environments. If a sensor malfunction or failure is detected the robot needs to disregard sensor readings until the sensor recovers. Having available prior information would allow the incorporation of sensor data only when the sensor is functioning correctly. Sensor faults could be detected if sensor values deviate greatly from prior values for large sections of the building.

Outdoors environments

This thesis has explored the concept of constructing informative Bayesian priors for indoors environments. An extension of this work would be to construct priors using satellite images or aerial photographs as a source of prior information. These could then be used to update maps produced by robots operating outdoors. This approach can further be extended to UAVs operating outdoors.

Bibliography

[1] Matlab toolbox monte carlo localisation. http://uk.mathworks.com/help/robotics/ug/monte-carlo-localization-algorithm.html. Accessed July 2016.

[2] SLAM code libraries and datasets. http://openslam.org. Accessed May 2015.

[3] Robot operating system (ROS). http://wiki.ros.org. Accessed May 2015.

[4] Turtlebot website. http://www.turtlebot.com/. Accessed October 2015.

[5] Microsoft Kinect coordinate spaces. https://msdn.microsoft.com/en-us/library/hh973078, 2015. Accessed October 2015.

[6] Microsoft Kinect skeletal tracking. https://msdn.microsoft.com/en-us/library/hh973074, 2015. Accessed October 2015.

[7] AEC Initiative Committee. AEC (UK) BIM Protocol. September 2012.

[8] C. Ah-Soon and K. Tombre. Architectural symbol recognition using a network of constraints. *Pattern Recognition Letters*, 22:231–248, 2001.

[9] S. Ahmed, M. Liwicki, M. Weber, and A. Dengel. Improved automatic analysis of architectural floor plans. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 864–869, 2011.

[10] S. Ahmed, M. Liwicki, M. Weber, and A. Dengel. Automatic room detection and room labeling from architectural floor plans. In *Proceedings of the IAPR International Workshop on Document Analysis Systems*, pages 339–343, 2012.

[11] S. Ahmed, M. Weber, M. Liwicki, C. Langenhan, A. Dengel, and F. Petzold. Automatic analysis and sketch-based retrieval of architectural floor plans. *Pattern Recognition Letters*, 35:91–100, 2014.

[12] P.F. Alcantarilla, L.M. Bergasa, and F. Dellaert. Visual odometry priors for robust EKF-SLAM. In *Proceedings of the International Conference on Robotics and Automation*, pages 3501–3506, 2010.

[13] F. Amigoni, A. Visser, and M. Tsushima. RoboCup 2012 Rescue simulation league winners. *RoboCup 2012: Robot Soccer World Cup XVI, Lecture Notes in Computer Science, Springer*, 7500:20–35.

[14] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2007.

[15] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on Particle Filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2002.

[16] S. Azar, L. Couvreur, V. Delfosse, B. Jaspart, and C. Boulanger. An agent-based multimodal interface for sketch interpretation. In *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, pages 488–492, 2006.

[17] C. Baden-Powell, J. Hetreed, and A. Ross. *Architect's Pocket Book*. Elsevier, 2011.

[18] T. Bailey, J. Nieto, and E. Nebot. Consistency of the FastSLAM algorithm. In *Proceedings of the International Conference on Robotics and Automation*, pages 424–429, 2006.

[19] T. Bailey. *Mobile Robot Localisation and Mapping in extensive outdoor environments*. Doctoral thesis, The University of Sydney, 2002.

[20] T. Bailey and H. Durrant-Whyte. Simultaneous Localization and Mapping: Part II. *IEEE Robotics and Automation Magazine*, 13:108–117, 2006.

[21] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF-SLAM Algorithm. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 3562–3568, 2006.

[22] B. Balaguer, S. Balakirsky, S. Carpin, and A. Visser. Evaluating maps produced by urban search and rescue robots: lessons learned from RoboCup. *Autonomous Robots*, 27:449–464, 2009.

[23] D. Ballard. Generalizing the Hough Transform to detect arbitrary shapes. *Pattern Recognition*, 13:111–122, 1981.

[24] D. Barnes, W. Maddern, and I. Posner. Exploiting 3D semantic scene priors for online traffic light interpretation. In *Proceedings of the Intelligent Vehicles Symposium*, pages 573–578, 2015.

[25] K.R. Beevers and W.H. Huang. Fixed-lag sampling strategies for Particle Filtering SLAM. In *Proceedings of the International Conference on Robotics and Automation*, pages 2433–2438, 2007.

[26] M. Bernard and K. Kondak. Autonomous transportation and deployment with aerial robots for Search and Rescue Missions. *Journal Field of Robotics, Wiley*, 28:914–931, 2011.

[27] J.L. Blanco, J. Gonzalez, and J.a. Fernandez-Madrigal. Optimal Filtering for Non-parametric Observation Models: Applications to Localization and SLAM. *The International Journal of Robotics Research*, 29:1726–1742, 2010.

[28] D. Burschka, J. Geiman, and G. Hager. Optimal landmark configuration for vision-based control of mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, pages 3917–3922, 2003.

[29] R. Chatila. Robot Mapping: An Introduction. *Robotics and Cognitive Approaches to Spatial Mapping, Springer Tracts in Advanced Robotics*, 38:9–12, 2008.

[30] A. Chatterjee and F. Matsuno. A Neuro-Fuzzy assisted Extended Kalman Filter-based approach for Simultaneous Localization and Mapping (SLAM) problems. *IEEE Transactions on Fuzzy Systems*, 15:984–997, 2007.

[31] X. Chen and J. Dorsey. Sketching Reality : Realistic Interpretation of Architectural Designs. *ACM Transactions on Graphics*, 27:1–21, 2008.

[32] F.D.K. Ching. *Building Construction Illustrated*. Wiley, 2008.

[33] M. Choi, R. Sakthivel, and W.K. Chung. Neural network-aided Extended Kalman Filter for SLAM problem. In *Proceedings of the International Conference on Robotics and Automation*, pages 1686–1690, 2007.

[34] H. Choset and K. Nagatani. Topological Simultaneous Localization and Mapping (SLAM): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17:125–137, 2001.

[35] P. Closas and C. Fern. Particle Filtering with adaptive number of particles. In *Procedings of the IEEE Aerospace Conference*, pages 1 – 7, 2011.

[36] T. Collins, J. Collins, and C. Ryan. Occupancy grid mapping: An empirical evaluation. In *Proceedings of the Mediterranean Conference on Control and Automation*, pages 1–6, 2007.

[37] P. Corke, R. Paul, W. Churchil, and P. Newman. Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation. In *Proceedings of the Intelligent Robots and Systems*, pages 2085–2092, 2013.

[38] A. Davids. Urban Search And Rescue robots: from tragedy to technology. *IEEE Intelligent Systems*, 17:81–83, 2002.

[39] J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. In *Proceedings of the International Conference on Machine learning*, pages 233–240. 2006.

[40] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1052–1067, 2007.

[41] A.J. Davison. Real-time Simultaneous Localisation and Mapping with a single camera. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1403–1410, 2003.

[42] P. de la Puente and D. Rodriguez-Losada. Feature based graph-SLAM in structured environments. *Autonomous Robots*, 37:243–260, 2014.

[43] K. Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. Wiley Interscience Series in Systems and Optimization, 2001.

[44] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[45] F. Dellaert. Factor graphs and GTSAM: A hands-on introduction. 2012.

[46] G. Dissanayake, H. Durrant-Whyte, and T. Bailey. A computationally efficient solution to the Simultaneous Localisation and Map building (SLAM) problem. In *Proceedings of the International Conference on Robotics and Automation*, pages 1009–1014, 2000.

[47] G. Dissanayake, S. Huang, Z. Wang, and R. Ranasinghe. A review of recent developments in Simultaneous Localization and Mapping. In *Proceedings of the IEEE International Conference on Industrial and Information Systems*, pages 477–482, 2011.

[48] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba. A solution to the Simultaneous Localization and Map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17:229–241, 2001.

[49] K. Dulal. Nepal earthquake 2015. `https://commons.wikimedia.org/wiki/File:Nepal_Earthquake_2015_05.jpg`. Acessed May 2015.

[50] H. Durrant-Whyte and T. Bailey. Simultaneous Localization and Mapping: Part I. *IEEE Robotics & Automation Magazine*, 13:99 – 110, 2006.

[51] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22:46–57, 1989.

[52] A. Eliazar and R. Parr. DP-SLAM 2.0. In *Proceedings of the International Conference on Robotics and Automation*, pages 1314–1320, 2004.

[53] A. Eliazar and R. Parr. DP-SLAM: Fast, robust Simultaneous Localization and Mapping without predetermined landmarks. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 1135–1142, 2003.

[54] R.M. Faragher, C. Sarno, and M. Newman. Opportunistic radio SLAM for indoor navigation using smartphone sensors. In *Proceedings of the Position, Location and Navigation Symposium*, pages 120–128, 2012.

[55] M.A. Fischler and R.C. Bolles. Random Sample Consensus: A Paradigm for model fitting with applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24:381–395, 1981.

[56] J. Folkesson and H. Christensen. Graphical SLAM - a self-correcting map. In *Proceedings of the International Conference on Robotics and Automation*, pages 383–390, 2004.

[57] J. Fossel, D. Hennes, D. Claes, S. Alers, and K. Tuyls. OctoSLAM: A 3D mapping approach to situational awareness of unmanned aerial vehicles. In *Proceedings of the International Conference on Unmanned Aircraft Systems*, pages 179–188, 2013.

[58] J. Fountain. Urban Search And Rescue: What is it? *Inside HART (Hazardous Area Response Team)*, pages 1–4, 2009.

[59] C. Georgiou, S. Anderson, and T. Dodd. Constructing contextual SLAM priors using architectural drawings. In *Proceedings of the International Conference on Automation, Robotics and Applications*, pages 50–56, 2015.

[60] C. Georgiou, S. Anderson, and T. Dodd. Constructing informative Bayesian map priors: a multi-objective optimisation approach applied to indoor occupancy grid mapping. *International Journal of Robotics Research*, 2017.

[61] B. Gerkey. gmapping ros package. `http://wiki.ros.org/gmapping`. Accessed May 2015.

[62] I. Giagkiozis, R.C. Purshouse, and P.J. Fleming. An overview of population-based algorithms for multi-objective optimisation. *International Journal of Systems Science*, 46:1572–1599, 2015.

[63] T. Gindele, S. Brechtel, J. Schroder, and R. Dillmann. Bayesian Occupancy grid Filter for dynamic environments using prior map knowledge. In *Proceedings of the Intelligent Vehicles Symposium*, pages 669–676, 2009.

[64] D. Greer, P.M. Kerrow, and J. Abrantes. Robots in Urban Search and Rescue operations. In *Proceedings of Australian Conference on Robotics and Automation*, pages 27–29, 2002.

[65] R. Grewe, M. Komar, A. Hohm, S. Lueke, and H. Winner. Evaluation method and results for the accuracy of an automotive occupancy grid. In *Proceedings of the IEEE International Conference on Vehicular Electronics and Safety*, pages 19–24, 2012.

[66] G. Grisetti. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In *Proceedings of the International Conference on Robotics and Automation*, pages 2432–2437, 2005.

[67] G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23:34–46, 2007.

[68] E. Guizzo. How Google's Self-Driving Car Works. `http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works`. Accessed October 2012.

[69] D. Hähnel, D. Fox, W. Burgard, and S. Thrun. A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. In *Proceedings of the Conference on Intelligent Robots and Systems*, 2003.

[70] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, pages 147–151, 1988.

[71] J.A. Hartigan and M.A. Wong. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28:100–108, 1979.

[72] J. Hartmann, D. Forouher, M. Litza, J.H. Klüssendorff, and E. Maehle. Real-time visual SLAM using FastSLAM and the Microsoft Kinect camera. In *Proceedings of the German Conference on Robotics*, pages 458–463, 2012.

[73] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, 31:647–663, 2012.

[74] D. Herath, S. Kodagoda, and G. Dissanayake. Simultaneous Localisation and Mapping: A stereo vision based approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 922–927, 2006.

[75] M. Himmelsbach, A. Mueller, T. Luettel, and H.J. Wuensche. LIDAR-based 3D object perception. In *Proceedings of the International Workshop on Cognition for Technical Systems*, 2008.

[76] H. Hodson. Robots to the rescue. *New Scientist*, 21:19 – 20, 2013.

[77] S. Holmes, G. Klein, and D.W. Murray. A Square Root Unscented Kalman Filter for visual monoSLAM. In *Proceedings of the International Conference on Robotics and Automation*, pages 3710–3716, 2008.

[78] D. Holz, C. Lorken, and H. Surmann. Continuous 3D sensing for navigation and SLAM in cluttered and dynamic environments. In *Proceedings of the International Conference on Information Fusion*, pages 1–7, 2008.

[79] D. Holz, D. Droeschel, S. Behnke, and S. May. *Fast 3D Perception for collision avoidance and SLAM in domestic environments*, chapter 4, pages 53–85. Mobile Robots Navigation, InTech, 2003.

[80] G.P. Huang, A. Mourikis, and S. Roumeliotis. Analysis and improvement of the consistency of extended Kalman filter based SLAM. In *Proceedings of the International Conference on Robotics and Automation*, pages 473–479, 2008.

[81] G.P. Huang, A.I. Mourikis, and S.I. Roumeliotis. On the complexity and consistency of UKF-based SLAM. In *Proceedings of the International Conference on Robotics and Automation*, pages 4401–4408, 2009.

[82] S. Huang and G. Dissanayake. Convergence and consistency analysis for Extended Kalman Filter based SLAM. *IEEE Transactions on Robotics*, 23:1036–1049, 2007.

[83] H.K. J, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. RoboCup Rescue: Search and Rescue in large-scale disasters as a domain for autonomous agents research. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 739–743.

[84] R. Jaulmes, E. Moliné, and J.O. Leclef. Towards a quantitative evaluation of Simultaneous Localization and Mapping methods. In *Proceedings of the National Conference on Control Architectures of Robots, France*, 2009.

[85] R. Juchmes, P. Leclercq, and S. Azar. A freehand-sketch environment for architectural design supported by a multi-agent system. *Computers and Graphics*, 29:905 – 915, 2005.

[86] S.J. Julier and J.K. Uhlmann. New extension of the Kalman filter to nonlinear systems. In *Proceedings of AeroSense*, pages 182–193. International Society for Optics and Photonics, 1997.

[87] C. Kim, R. Sakthivel, and W.K. Chung. Unscented FastSLAM: A robust and efficient solution to the SLAM problem. *IEEE Transactions on Robotics*, 24: 808–820, 2008.

[88] A. Kleiner and C. Dornhege. Real-Time Localization and Elevation Mapping within Urban Search and Rescue scenarios. *Journal of Field Robotics, Wiley*, 24:723–745, 2007.

[89] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard. Large scale graph-based SLAM using aerial images as prior information. *Autonomous Robots*, 30:25–39, 2011.

[90] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf. A flexible and scalable SLAM system with full 3D motion estimation. In *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 155–160, 2011.

[91] K. Konolige, E. Marder-eppstein, B. Marthi, and W. Garage. Navigation in hybrid metric-topological maps. In *Proceedings of the International Conference on Robotics and Automation*, pages 3041–3047, 2011.

[92] A. Koutamanis. Digital architectural visualization. *Automation in Construction*, 9:347–360, 2000.

[93] V. Kumar, D. Rus, and S. Singh. Robot and Sensor Networks for First Responders. *IEEE Pervasive Computing*, 3:24–33, 2004.

[94] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g 2 o: A general framework for graph optimization. In *Proceedings of the International Conference on Robotics and Automation*, pages 3607–3613, 2011.

[95] T. Lemaire, C. Berger, I.K. Jung, and S. Lacroix. Vision-Based SLAM: Stereo and monocular approaches. *International Journal of Computer Vision, Springer*, 74:343–364, 2007.

[96] Y. Li, E.B. Olson, and A. Arbor. Extracting general-purpose features from LIDAR data. In *Proceedings of the International Conference on Robotics and Automation*, pages 1388 – 1393, 2010.

[97] Y. Liu and S. Thrun. Results for outdoor-SLAM using Sparse Extended Information Filters. In *Proceedings of the International Conference on Robotics and Automation*, pages 1227–1233, 2003.

[98] P. Lourenço, P. Batista, P. Oliveira, C. Silvestre, and C.L. Philip Chen. Sensor-based globally asymptotically stable range-only Simultaneous Localization and Mapping. In *Proceedings of the IEEE Conference on Decision and Control*, pages 5692–5697, 2013.

[99] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, 4(4):333–349, 1997.

[100] S. Macé, H. Locteau, E. Valveny, and S. Tabbone. A system to detect rooms in architectural floor plan images. In *Proceedings of the IAPR International Workshop on Document Analysis Systems*, pages 167–174. 2010.

[101] W. Maddern, G. Pascoe, and P. Newman. Leveraging experience for large-scale LIDAR localisation in changing cities. In *Proceedings of the International Conference on Robotics and Automation*, pages 1684–1691, 2015.

[102] Maritime & Coastguard Agency. Search and Rescue Framework for the United Kingdom of Great Britain and Northern Ireland. 2008.

[103] M.C. Martin and H.P. Moravec. Robot Evidence Grids. *Carnegie Mellon University Tech Report*, 1996.

[104] R. Martinez-Cantin and J. Castellanos. Unscented SLAM for large-scale outdoor environments. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 3427–3432, 2005.

[105] M. Mazuran, G.D. Tipaldi, and C. Stachniss. A statistical measure for map consistency in SLAM. In *Proceedings of the International Conference on Robotics and Automation*, pages 3650–3655, 2014.

[106] D. Meyer-Delius, M. Beinhofer, and W. Burgard. Occupancy Grid models for robot mapping in changing environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2024–2030, 2012.

[107] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics, Wiley*, 29:832–841, 2012.

[108] A. Milstein. *Occupancy Grid Maps for Localization and Mapping*, chapter 19, pages 381–408. Motion Planning, InTech, 2005.

[109] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0 : An improved Particle Filtering algorithm for Simultaneous Localization and Mapping that provably converges. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 1151–1156.

[110] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: a factored solution to the Simultaneous Localization and Mapping problem. In *Proceedings of the National Conference on Artificial intelligence, Canada*, pages 593–598, 2002.

[111] R.R. Murphy. Trial by Fire. *IEEE Robotics and Automation Magazine*, 11:50–61, 2001.

[112] R.R. Murphy and S. Stover. Rescue Robots for mudslides: A descriptive study of the 2005 La Conchita mudslide response. *Journal Field of Robotics, Wiley*, 25:3–16, 2008.

[113] K. Nagatani, S. Kiribayashi, Y. Okada, S. Tadokoro, T. Nishimura, and T. Yoshida. Redesign of rescue mobile robot Quince: Toward emergency response to the nuclear accident at Fukushima Daiichi nuclear power station on March 2011. In *Proceedings of the IEEE International SYmposium on Safety, Security and Rescue Robotics*, pages 13–18, 2011.

[114] K. Nagatani, Y. Okada, N. Tokunaga, S. Kiribayashi, K. Yoshida, K. Ohno, E. Takeuchi, S. Tadokoro, H. Akiyama, I. Noda, T. Yoshida, and E. Koyanagi. Multirobot exploration for Search and Rescue missions : A report on map

building in RoboCupRescue 2009. *Journal of Field Robotics, Wiley*, 28:373–387, 2011.

[115] A. Napier, P. Corke, and P. Newman. Cross-Calibration of Push-Broom 2D LIDARs and Cameras In Natural Scenes. In *Proceedings of the International Conference on Robotics and Automation*, pages 3679–3684, 2013.

[116] P. Newman. SLAM-loop closing with visually salient features. In *Proceedings of the International Conference on Robotics and Automation*, pages 635–642, 2005.

[117] P. Ngatchou, A. Zarei, and M. El-Sharkawi. Pareto multi objective optimization. In *Proceedings of the International Conference on Intelligent Systems Application to Power Systems*, pages 84–91, 2005.

[118] R. Noack. *Converting CAD drawings to product models*. Licentiate thesis, Royal Institute of Technology Stockholm, Sweden, 2001.

[119] S.H. Or, K.H. Wong, Y.K. Yu, and M.M.Y. Chang. Highly automatic approach to architectural floorplan image understanding & Model Generation. *Pattern Recognition*, 2005.

[120] Oxford Mobile Robotics Group. RobotCar UK. `http://mrg.robots.ox.ac.uk/robotcar/index.html`. Accessed September 2013.

[121] O. Ozisik and S. Yavuz. An Occupancy Grid based SLAM method. In *Proceedings of the International Conferemce on Computational Intelligence for Measurement Systems And Applications*, pages 14–16, 2008.

[122] A.K. Pandey, K.M. Krishna, and H. Hexmoor. Feature chain based occupancy grid SLAM for robots equipped with sonar sensors. In *Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 283–288, 2007.

[123] M. Parsley and S. Julier. Towards the exploitation of prior information in SLAM. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 2991–2996, 2010.

[124] M. Parsley and S. Julier. Exploiting prior information in GraphSLAM. In *Proceedings of the International Conference on Robotics and Automation*, pages 2638–2643, 2011.

[125] L. Paz, P. Jensfelt, J. Tardos, and J. Neira. EKF SLAM updates in O(n) with Divide and Conquer SLAM. In *Proceedings of the International Conference on Robotics and Automation*, pages 1657–1663, 2007.

[126] I.T. Phillips and A.K. Chhabra. Empirical performance evaluation of graphics recognition systems. *IEEE Transactions Pattern Analysis Machine Intelligence*, 21:849–870, 1999.

[127] K. Pirker, M. Rüther, H. Bischof, and G. Schweighofer. Fast and accurate environment modeling using three-dimensional occupancy grids. pages 1134–1140, 2011.

[128] D.M.W. Powers. Evaluation: From precision, recall and F-Measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2:37–63, 2011.

[129] S. Shen, N. Michael, and V. Kumar. Autonomous multi-floor indoor navigation with a computationally constrained MAV. In *Proceedings of the International Conference on Robotics and Automation*, pages 20–25, 2011.

[130] S. Shen, N. Michael, and V. Kumar. Autonomous indoor 3D exploration with a micro-aerial vehicle. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 9–15, 2012.

[131] C. Stachniss, D. Hähnel, W. Burgard, and G. Grisetti. On actively closing loops in grid-based FastSLAM. *Advanced Robotics*, 19:1059–1079, 2005.

[132] S. Takezawa, D. Herath, and G. Dissanayake. SLAM in indoor environments with stereo vision. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1866–1871, 2004.

[133] The National Search And Rescue Council. National Search and Rescue manual. *Australian Search And Rescue Manuals*, 2011.

[134] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge Massachussets, 2006.

[135] S. Thrun. Particle Filters in Robotics. In *Proceedings of Uncertainty in AI*, 2002.

[136] S. Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15:111–127, 2003.

[137] S. Thrun and Y. Liu. Multi-robot SLAM with Sparse Extended Information Filters. In *Proceedings of the International Symposium of Robotics Research*, pages 254–266, 2005.

[138] S. Thrun and M. Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006.

[139] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*, 128:99–141, 2001.

[140] S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and A.Y. Ng. Simultaneous Localization and Mapping with Sparse Extended Information Filters. *The International Journal of Robotics*, 23:693–716, 2004.

[141] Tim Bailey. EKF-SLAM version 2. `http://www-personal.acfr.usyd.edu.au/tbailey/software/slam_simulations.htm`, 2013. Accessed December 2012.

[142] R. Triebel, H. Grimmett, R. Paul, and I. Posner. Introspective active learning for scalable semantic mapping. In *Proceedings of the Workshop on Active Learning in Robotics: Exploration, Curiosity and Interaction. Robotics Science and Systems*, 2013.

[143] University of Freiburg. Robotics datasets. `http://www2.informatik.uni-freiburg.de/~stachnis/datasets.html`. Accessed May 2015.

[144] U.S. Department of Homeland Security. Structure fire response times. *Topical Fire Research Series*, 5:5–10, 2006.

[145] R. Van Den Boomgaard and R. Van Balen. Methods for fast morphological image transforms using bitmapped binary images. *CVGIP: Graphical Models and Image Processing*, 54:252–258, 1992.

[146] R. Van Der Merwe and E. Wan. The square-root Unscented Kalman Filter for state and parameter-estimation. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 3461–3464, 2001.

[147] M. van Zomeren. Search methods in USAR. Technical report, Delft Insitute of Technology, Delft, 2008.

[148] A. Vatavu, R. Danescu, and S. Nedevschi. Environment perception using dynamic polylines and particle based occupancy grids. In *Proceedings of the International Conference on Intelligent Computer Communication and Processing*, pages 239–244, 2011.

[149] M.R. Walter, R.M. Eustice, and J.J. Leonard. Exactly Sparse Extended Information Filters for feature-based SLAM. *The international journal of robotics research*, 26(4):335–359, 2007.

[150] N. Wang, S. Ma, B. Li, M. Wang, and M. Zhao. A hybrid map representation for Simultaneous Localization and Mapping of the internal ruins environment. In *Proceedings of the IEEE International Conference on Information and Automation*, pages 1060–1065, 2013.

[151] J. Weingarten and R. Siegwart. EKF-based 3D SLAM for structured environment reconstruction. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 3834–3839, 2005.

[152] C. Weyers and G. Peterson. Improving occupancy grid FastSLAM by integrating navigation sensors. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 859–864, 2011.

[153] B. Williams and I. Reid. On combining visual SLAM and visual odometry. In *Proceedings of the International Conference on Robotics and Automation*, pages 3494–3500, 2010.

[154] C. Wright, A. Johnson, A. Peck, Z. McCord, A. Naaktgeboren, P. Gianfortoni, M. Gonzalez-Rivero, R. Hatton, and H. Choset. Design of a modular snake robot. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 2609–2614, 2007.

[155] K.M. Wurm and G. Grisetti. Bridging the gap between feature and grid based SLAM. *Robotics and Autonomous Systems*, 58:140–148, 2010.

[156] N. Yang, B. Ding, and J. Zhang. Improved Sampling Importance Resampling Particle Filter optimized by tabu search. In *Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence*, pages 359–363, 2010.

[157] T. Yaqub and J. Katupitiya. Laser scan matching for measurement update in a Particle Filter. In *Proceedings of the International Conference on Advanced Intelligent Mechatronics*, pages 1–6, 2007.

[158] X. Yin, P. Wonka, and A. Razdan. Generating 3D building models from architectural drawings: A Survey, 2009.