Experience Based Action Planning for Environmental Manipulation in Autonomous Robotic Systems

Richard Alan Redpath

MPhil

University of York
Electronic Engineering

March, 2019

Abstract

The ability for autonomous robots to plan action sequences in order to manipulate their environment to achieve a specific goal is of vital importance for agents which are deployed in a vast number of situations. From domestic care robots to autonomous swarms of search and rescue robots there is a need for agents to be able to study, reason about, and manipulate their environment without the oversight of human operators. As these robots are typically deployed in areas inhabited and organised by humans it is likely that they will encounter similar objects when going about their duties, and in many cases the objects encountered are likely to be arranged in similar ways relative to one another. Manipulation of the environment is an incredibly complex task requiring vast amounts of computation to generate a suitable state of actions for even the simplest of tasks. To this end we explore the application of memory based systems to environment manipulation planning. We propose new search techniques targeted at the problem of environmental manipulation for search and rescue, and recall techniques aimed at allowing more complex planning to take place with lower computational cost. We explore these ideas from the perspective of autonomous robotic systems deployed for search and rescue, however the techniques presented would be equally valid for robots in other areas, or for virtual agents interacting with cyber-physical systems.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to begin by thanking my parents, Jacqueline and Richard, and my sister, Susan. Your continued support throughout my education and early career has been a constant source of motivation as I have continued on my journey, ultimately culminating here. Without the motivation, drive, and confidence that you have all instilled in me over the years I undoubtedly would not be in the position I currently am and for that I am eternally grateful. You have always all been so proud of what I have achieved and I can not put in to words how lucky that makes me feel each and every day.

Next, I would like to thank my supervisors; Jon Timmis and Martin Trefzer. I could not have asked for a better pairing to go on this journey with and I will be forever in your debt. You have both kept me focused on the end goal, while allowing me to explore ideas, develop new skills, grow as a person, and finally get to the point I am today. You have both taught me so much over the last few years and have proven to be great friends as well as incredible supervisors.

I'd also like to thank my friends and colleagues within the department. James, Naomi, Edgar, Alan, Kieran, Becky, Andy, Stu, Nils, Matt, Sam, Nick, and countless others besides. You have all kept me on just the right side of sane since I started in the department. You have all been there for me when I needed anything from a discussion about work or coffee to unwind and for that I can't thank you all enough.

Last but not least I'd like to thank everybody in the department and wider university who helped make this come true and who I have been able to share the last few years with. The combined ability of every member of the department to stay so close knit and friendly has made my time in the department and absolute pleasure and a part of my life I will never forget.

I have undoubtedly missed a few people out but I hope you will all know who you are. I can't believe how fantastically lucky I have been to have this opportunity and to share it with such amazing people. Thank you all.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

Chapter 1

Introduction

1.1 Introduction

Autonomous robotic systems have great promise in exploring areas which may prove dangerous, inaccessible, or inhospitable to humans due to hazardous environments. Often, these hazardous environments are the result of destructive events which may leave survivors trapped in debris, resulting in a situation where search and rescue operations are launched in order to discover and free them. While remote control systems exist for human operators to guide a robot through the wreckage these solutions are not always viable options as a result of communication difficulties.

Even in situations where communication is not an issue it is not always feasible for a single robot to remove all obstructions that may be encountered during the search process. To combat this, swarm robotics could be utilised in order to increase the number of agents in a given scenario, not only allowing the location to be searched more efficiently, but also increasing the range of actions that could be undertaken to aid exploration.

In this work we do not explicitly consider the case of multiple robots, but the techniques used are motivated by their applicability to multi-robot systems. A number of techniques exist for both multi-robot and human-robot collaboration and communication, a brief review of which can be found in (Gildert et al., 2018). To this end we focus on the planning process which would allow a collective to discover action sequences which can be used to clear an obstruction, allowing previously inaccessible locations to be reached.

Traditionally, swarm robotics has considered collectives of small, relatively simple, independent agents which produce a macro effect by following a set of rules which rely only on local knowledge and communication. These approaches often take inspiration from biological collectives, where the behaviour of individuals can be explained using very simple rules, yet complex behaviours occur at the level of the collective.

In (Sharkey, 2007) this traditional approach is referred to as “Nature-inspired Minimalist

Swarm Robotics", and is characterised by a minimalist approach with nature-inspired constraints. A more relaxed approach is referred to as "Practical Minimalist Swarm Robotics" which takes the same minimalist approach to system design, but is not constrained by biological influences. A final approach is simply named "Scalable Swarm Robotics", and is characterised simply by decentralised design approaches which are independent of swarm size, giving rise to inherent scalability.

As computation becomes cheaper, smaller, and more energy efficient, the ability for individuals of a swarm to exhibit complex individual behaviours, while displaying simple inter-agent behaviours, becomes more tractable. This would allow for collectives which are more reminiscent of higher-order mammals than of the simple biological agents which formed the basis of swarm robotics research.

Given the availability of cheap, power-efficient, high performance computing resources, it is possible for individuals to reason more deliberatively about their environment. It is now possible for simple agents to run on-board simulations of their environment, predicting the outcomes of events before actions are executed, or detecting discrepancies in expected behaviours. Examples of this include Winfield's consequence engine (Winfield et al., 2014),and Millard's internal simulations for fault detection (Millard et al., 2014).

This can be extended to physics simulations in which agents can predict the state of the world as a result of their potential interactions with objects in their environment. This ability for robots to "imagine" the outcomes of their actions opens up an exciting opportunity for small, cheap, energy-efficient robots to plan action sequences in order to overcome previously unseen problems.

Although it is now computationally tractable to perform physics simulations onboard a mobile robot the search space of possible action sequences is too large to be searched naively, requiring approaches which can quickly disregard large areas of the search space which are unlikely to contain a solution.

The work and proposed solutions we present here are guided by the idea of robots "imagining" the consequences of actions, drawing parallels between this and the way collaborative action planning takes place in natural collectives. While large swarms often behave in a completely decentralised manner this is far less common in creatures which exhibit more complex behaviours, and which typically consist of small enough group sizes that social hierarchies can form and leadership can occur within the group.

If we look to humans as an inspiration of how collectives work to solve a problem we see that leadership is often dynamic and depends on the problem being faced. Due to the range of experiences that each member of a group might have the most suitable leader of a group will likely change based on who has the most relevant knowledge for the current situation. It is from this observation that we propose a system in which dynamic leadership can occur in a swarm or multi-robot system, and present initial work demonstrating how the objects in the world could be represented to allow for experience based action planning.

1.2 Contributions and Aims

In this work we explore a means of representing objects that a robot has encountered in a way which allows for efficient recall of objects which behave in similar ways when a robot interacts with them. It is our hope that the results presented here could be built upon by future work, allowing search processes to be augmented with a memory system, allowing for more sophisticated search techniques to be applied to environmental manipulation tasks.

The memory based system used to augment the search process deviates from typical machine learning style object recognition, which is often based on vision alone, to increase knowledge transferability regardless of visual similarities of objects. A great deal of work has been completed in the fields of Simultaneous Localisation And Mapping (SLAM) (Smith et al., 1990), image segmentation, and more recently into Object Based SLAM. As a result of the substantial body of both historic and ongoing work in these areas we do not consider them explicitly, instead assuming that an Object Based SLAM system is available to produce maps of the environment.

With these points in mind we aim to address the following claim:

> By using observations of interactions with objects in the world a low dimensional representation can be produced which allows for efficient recall of objects which have been observed to react similarly to one another.

It has recently been shown that MCTS can be augmented with a memory system in order to improve search efficiency, for instance in AlphaGo (Silver et al., 2016). It has also been shown that MCTS can be applied to single agent action planning in order to reach a goal where there are constraints on available search time. To the best of our knowledge, however, no work has previously been done to address robotics applications of MCTS for environmental interaction.

Given a suitable means to recall scenes which have already been encountered we believe that MCTS could prove to be an incredibly powerful tool for solving complex action planning problems. We present how such a system could be structured in chapter 3.

In this work we provide a number of contributions toward this solution, and evaluate the hypothesis stated above. Specifically:

- We present an overall architecture which could be used to develop an autonomous multi-agent system capable of capitalising on the experiences of each individual within it to develop action plans for the collective in order to overcome an obstacle

- We develop a method of recalling objects based on the way they respond to interactions by an agent, increasing the transferability of previously gained knowledge which relates to those objects
- We provide a dataset of observed object interactions which can be used by the wider research community as a basis for further work

1.3 Thesis Structure

The rest of this thesis is structured as follows.

In chapter 2 we review the current state of the art in the relevant literature. We discuss what we believe to be shortcomings in the state of the art and identify the areas which we believe could provide potential for current and future work.

In chapter 3 we discuss a proposed architecture which encompasses aspects of planning, recall, and multi agent communication. We provided an overview of proposed techniques which would work together to form a cohesive system, providing context and motivation for the work we have carried out.

In chapter 4 we argue that traditional object recall techniques are not entirely applicable to accelerating environmental interaction planning and do not lend themselves to knowledge transfer as well as our proposed methods. We then present work in producing an embedding space which is better suited to this task.

In chapter 5 we summarise the work completed, and evaluate our contributions to the state of the art.

Chapter 2

Background & Related Work

In the following sections we will provide background for a number of existing techniques and review some of the existing literature surrounding the areas of interest within our proposed system. Where relevant each section includes a brief description of where we feel the techniques or concepts could be useful. More details of where we believe these techniques to be useful is provided in chapter 3.

We begin by describing Simultaneous Localisation and Mapping (SLAM). This is the process of a robot producing a map of its environment, while simultanously tracking its own location within that map. We explore the history of a number of SLAM techniques, and discuss some more recent developments in the field which allow objects to be detected in the produced map, resulting in a model of the environment which can be more freely reasoned about. We finish off the discussion of SLAM by presenting a number of techniques which allow SLAM to be deployed in multi-agent systems.

We then provide an overview of Markov Decision Processes (MDPs). These are a class of problem which assume that the best action to take from any given state is determined entirely by the state itself. The problem of planning action sequences to manipulate the environment a robot is in can be seen as a MDP and as such a large amount of relevant work has been done in the field. Here we present only the work that we believe to be directly relevant to understanding the system proposed in chapter 3.

Next, we cover dimensionality reduction techniques. These techniques allow complex data to be stored in far simpler forms, allowing for more efficient processing in many machine learning contexts. We present a number of techniques which we later apply in chapter 4.

Finally, we present literature related to the recall of collections of objects (referred to as "scenes" in this work). We discuss our motivation for the techniques we propose, and highlight some relevant work from the field of machine learning.

2.1 Simultaneous Localisation and Mapping (SLAM)/Object based SLAM

In this section we describe the problem of SLAM, and a number of approaches which could be used to build up a map for a swarm of robots to use with the techniques developed later. Typically the driving force behind automated exploration using robots is that we are completely unfamiliar with the environment being explored and that the area in question is either dangerous, inaccessible, or inhospitable to humans.

SLAM presents a number of problems for robots which are required to explore these areas. When a robot is attempting to build a 3D map it is necessary for it to produce a graph of connected vertices which represent features in the environment. We can therefore observe that it must be able to locate and position landmarks (vertices) in this map, which is a relatively simple task if the robot can be sure of its location and has reliable range and bearing information of identified landmarks.

Often, the purpose of robot exploration is either for a robot to be able to act independently for extended periods of time in a potentially dynamic environment, or to gain information about an environment which is not accessible to humans. Generally it is not possible for a robot to be sure of its own location, either absolutely or in relation to a particular landmark or set of landmarks due to sensor noise. It is also not possible to assume that the robot can track its position by simply integrating its intended movements through space as environmental factors (such as wind, water currents or wheel slippage) mean that these estimates of movement are often incorrect.

We are therefore left with a situation in which we need to estimate the location of landmarks in the environment from noisy sensor readings which have been taken at a noisy, estimated location while maintaining as much accuracy as possible. Fortunately we can use our estimated landmark locations to gain a more accurate estimation of our location which in turn makes later landmark estimates more accurate. This observation was first made in a seminal work by Smith and Cheeseman (Smith & Cheeseman, 1986) and has formed the basis of most of the successful SLAM systems in use today.

Maps generated using SLAM based techniques can be augmented with information about the objects which can be detected within the scene. This augmentation process is often known as either semantic SLAM or, more recently, object(-based) SLAM. This section will begin by briefly reviewing traditional SLAM, which produces static meshes, before moving on to review some of the more recent literature which augments these meshes with semantic information from a database of known objects. This identification of objects allows us to reason about the environment and improve our knowledge of the world and its contents, including physical attributes and potentially useful actions which can be performed on the object.

2.1.1 Simultaneous Localisation and Mapping (SLAM)

The complexity of the SLAM problem arises as a result of uncertainty in both motion data and observational sensor data. Our robot starts in an arbitrary location and can estimate how much it moves, however the noise in this movement data means that as time progresses our robot would become less and less certain of its actual location. At the same time as moving through its environment our robot attempts to build up a map of features in the environment based on data from its sensors, such as a cameras or laser scanners, basing its estimate of feature locations in 3D space on its own estimated position and its sensor readings. By using each of these estimates to improve the quality of each other and repeatedly adding information gained from motion estimates and feature detection we are able to iteratively map and localise far more accurately than would be possible by attempting to solve each of these problems independently.

More formally, taking a probabilistic view, we are aiming to maximise

$$p(m_T, l_T | o_{1..T})$$

where m_T and l_T represent our current map and location, and o_t represents the observations we made of our environment at time t. One of the ways in which SLAM implementations can be classified is on the way they represent the world, often either representing the world as a grid, or as a continuous space containing points of interest.

When using a grid based implementation our map is simply a 2D or 3D grid representing a discretised world with each cell being updated with our belief about its contents, for example whether or not a cell contains something and, if so, its colour.

In continuous space implementations the world is represented as a collection of landmarks with much the same information as the grid based methods, with the exception that each landmark must also have a location and a record of the uncertainty associated with that location. For our purposes a continuous space implementation would be far more suitable as it allows for more fine-grained information about the location of features in space, thus increasing the quality of any matching processes which will rely on the relative locations of those features.

A seminal work in the SLAM field (Smith et al., 1990) proposed the use of an Extended Kalman Filter (EKF) to track the estimated location and relative error of each environmental feature and the robot location. New information from motion and observational sensors can be incorporated gracefully using this method, however the computational cost of the EKF can quickly become prohibitive in a real world environment and the Gaussian noise assumption required to make use of EKFs can often be a limiting factor in EKF based SLAM methods. EKF SLAM's computational complexity and Gaussian noise assumption are well known to be limiting factors in the robotics community and have been noted by a number of authors

(Montemerlo et al., 2003; Bailey et al., 2006).

Since the early SLAM systems such as the one presented in (Leonard & Durrant-Whyte, 1991) a great deal of work has been done to improve the quality of SLAM algorithms, some attempting to improve the computational complexity and reliability of EKF based techniques (Paz et al., 2008; Guivant & Nebot, 2001). Others have attempted to develop completely novel algorithms such as FastSLAM (Montemerlo et al., 2003) and Decoupled Stochastic Mapping (Leonard & Feder, 2000) which use different statistical techniques such as particle filters (Howard, 2006; Pei et al., 2014) or correct earlier errors as new data is collected (Engelson & McDermott, 1992).

Object Based SLAM

A recent modification to traditional SLAM techniques is SLAM++(Salas-Moreno et al., 2013), described by the authors as ‘SLAM at the level of objects’. As noted by the authors, many scenes in the modern world consist of ‘repeated, domain-specific objects and structures’. These can be detected during the SLAM process to produce a map of a scene which consists of distinct objects rather than the indiscriminate point cloud usually produced by SLAM algorithms. Without this point cloud segmentation it becomes considerably more difficult to reason about the world our robot is in and our understanding of the physical properties of our robot’s environment is highly limited.

At the time of writing the implementation used in (Salas-Moreno et al., 2013) relies on a GPU implementation to provide sufficient computational power to perform in-the-loop object detection, however as a new paradigm for the SLAM problem there will likely be much improvement in the speed and accuracy of this style of technique in the coming years, with algorithms becoming better suited to a wider array of tasks and environments.

It is also worth noting that, at present, the objects which can be detected by SLAM++ must be scanned offline before the system is used for mapping and repeated structures cannot be detected in an online manner, however the structure of the algorithm does not inherently restrict this extension. The current use of Drost et al’s. Point Pair Features (Drost et al., 2010) would likely make this extension expensive to perform in an online manner. In a complete system an extended SLAM++ system would be highly beneficial to allow agents to gain knowledge of the world during exploration and to predict and learn the physical properties of objects discovered in novel environments, although new objects could be learnt in an offline manner during a ‘reflection period’ between explorations.

More recently the requirement for a GPU implementation for a small database of objects has been overcome in (Gálvez-López et al., 2015) where the authors present a monocular SLAM system which is able to detect objects from a 500 item database in real time on a standard CPU. Given the rate at which progress has been made in the field of object based SLAM it is not unreasonable to believe that, in the near future, techniques which perform object based SLAM will be efficient enough to be used on mobile devices. Low powered, high speed, multi-core processors are already commonplace today with many of them also incorporating graphics pipelines and support for OpenCL allowing for efficient implementation of parallelizable techniques.

As the field of robotics has progressed and the consumer market has provided greater demand for augmented reality applications there has been a growing need for systems which are not only able to build up sparse maps of their environment but also to understand and reason about them (Conditt, 2015). The particular field of this we are interested in and will investigate in greater detail is the augmentation of a SLAM map with higher level identified objects.

The shear magnitude of improvement in object based SLAM systems since the publication of SLAM++ shows the level of interest in making use of these techniques. Recently a number of techniques have been presented which allow object level SLAM to take place using a single RGB camera (Davison et al., 2007) by using distinctive visual features, detected using machine vision techniques such as SIFT (Lowe, 1999) or SURF (Bay et al., 2006), and which can be used in pseudo-realtime with databases of hundreds of objects on a standard CPU (Pillai & Leonard, 2015; Gálvez-López et al., 2015). Each of these methods shows great improvement on the results presented in (Salas-Moreno et al., 2013) increasing both the speed and practicality of object based SLAM.

In the interest of making our system applicable to applications involving cheaper, lighter, and more manoeuverable robots we assume the use of one of these monocular visual SLAM systems. While we will work under this assumption it should be noted that this should not restrict the applicability of any developed techniques to any other form of sensor (such as RGB-D cameras, thermal imaging cameras or laser scanners) as long as a technique can be implemented which reliably identifies objects in the robot's environment.

Distributed SLAM (DSLAM)

While traditional SLAM typically takes place on a single agent, work has recently taken place to distribute the mapping process across a swarm of agents, not only increasing mapping speed but also potentially allowing access to areas which may have otherwise been inaccessible by allowing the use of a heterogeneous agents in the swarm. In a DSLAM system each agent explores it's own local environment and shares the knowledge it has discovered with other agents in the swarm.

Each of these individually shared maps are then combined using Data Fusion algorithms, resulting in a single map containing data from multiple agents in the same environment.

As with most swarm algorithms there are both fully distributed algorithms and centralised algorithms for the DSLAM problem. In both of these cases the aim is to solve the same problem and the choice of using a distributed or centralised technique can be decided by the need for live human interaction or feedback and the computational requirement of the technique used.

One of the more well known D-SLAM techniques is DDF-SAM (Cunningham et al., 2010). DDF-SAM breaks the problem of D-SLAM in to three components: local optimisation to produce a per-agent graph of the environment, a communication module to propagate these condensed graphs across the swarm, and a neighbourhood optimiser to combine multiple graphs in to a single map. In (Carlone et al., 2010) the authors present an alternative method which makes use of particle filter algoritms to fuse the maps produced by robots in a collective, without requiring and estimation of their initial locations. (Andre et al., 2014) provides an overview of several implementations of D-SLAM algorithms which have been released for the Robot Operating System (ROS).

2.2 Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) are a class of decision problem in which we assume that the optimal action at any given time is decided only by the current world state, without considering the history of the world or the history of our actions (Bellman, 1957). Known as the Markov assumption this holds for the situations we are interested in, any actions we may have already performed have changed the state of the world and our available resources and are irreversible. As such we gain no benefit from considering how the world used to be or how much power we used to have, only the way that we and the world currently are. A Markov decision problem can be fully defined by 4 components:

S	: A set of world states
A	: A set of actions
$T(s, a, s')$	: A probability function over states, actions and next states
$R(s)$	: A reward function which provides feedback about the current state

Given a model containing these 4 components we aim to define an action policy, $\pi(s)$, which provides us with the best expected action to take in any provided world state.

In MDPs as defined above we work under the assumption that we can know the current world state with absolute certainty. This is often not the case in real world scenarios as we are usually unable to observe the entire world at once, and in a dynamic environment the state of the world which is currently out of our field of perception is unlikely to remain constant, giving rise to the aptly named concept of Partially Observable Markov Decision Processes (POMDPs) (Brechtel et al., 2013). This extension requires the addition of an observation model, $O(s, o)$, to our definition of MDPs which provides the probability of making observation o while the world is in state s. From this observation model it is possible to predict the world state that we are currently in based on the observations we can currently make, however there is an inherent level of uncertainty that we must take into consideration.

A great deal of work has been done in the field of solving POMDP planning problems including techniques which incorporate reinforcement learning (Wang et al., 2012), value iteration (Brechtel et al., 2013) and Monte-Carlo tree search methods (Silver & Veness, 2010) with several deployed examples of POMDP planning being performed on robots in real world scenarios such as (Shaojun, 2014). The complexity of POMDPs can vary widely with an investigation into why some POMDPs appear easier to generate solutions for being presented in (Hsu et al., 2007).

2.2.1 Monte-Carlo Tree Search (MCTS)

MCTS is a method for planning action sequences which has gained popularity in MDPs and POMDPs (Silver & Veness, 2010; Browne et al., 2012) lately due to its applicability in artificial intelligence for gaming. Tree based methods for planning typically work in a similar manner, with the world state being represented by a node in a search tree and the links between nodes representing state altering actions, producing a new world state as a result.

As an example consider attempting to plan a sequence of moves to solve the '8-puzzle'. In this puzzle we have a 3x3 grid containing 8 numbered tiles. We can use the remaining empty cell to move tiles around the grid by shifting them in 1 of the 4 cardinal directions. The goal is to move the tiles around the grid so that the empty space is in the lower right corner and the tiles are ordered from 1-8 reading left-to-right, top-to-bottom. When planning move sequences it is often more intuitive to consider moving the space in one of the available directions with the side effect of moving a tile in the opposite direction. This is a commonly used convention when discussing the 8-puzzle and is used here.

Traditional tree search methods explore trees in either a depth first or breadth first manner, either expanding child nodes immediately as they are created or expanding nodes row by row, gradually deepening the tree. In a number of cases we can find ourselves in infinitely deep trees, for example when an action can restore the world to a state which has already been visited in a node's history, so a breadth first search is often used to ensure we do not fall into these infinite expansions.

As an example consider a grid in the configuration of the root node shown in Figure 2.1. From here we have 4 possible moves which would move the empty space in each of the 4 cardinal directions, in turn moving a tile in the opposite direction. We simulate each of these 4 moves to give us a new world state without making a move in the real world. This process then continues on each of the modified states, expanding each branch of the tree until we generate a node with the target state. When we reach a goal state we have found a sequence of actions which will solve the puzzle from a given state by following the branches taken from the root

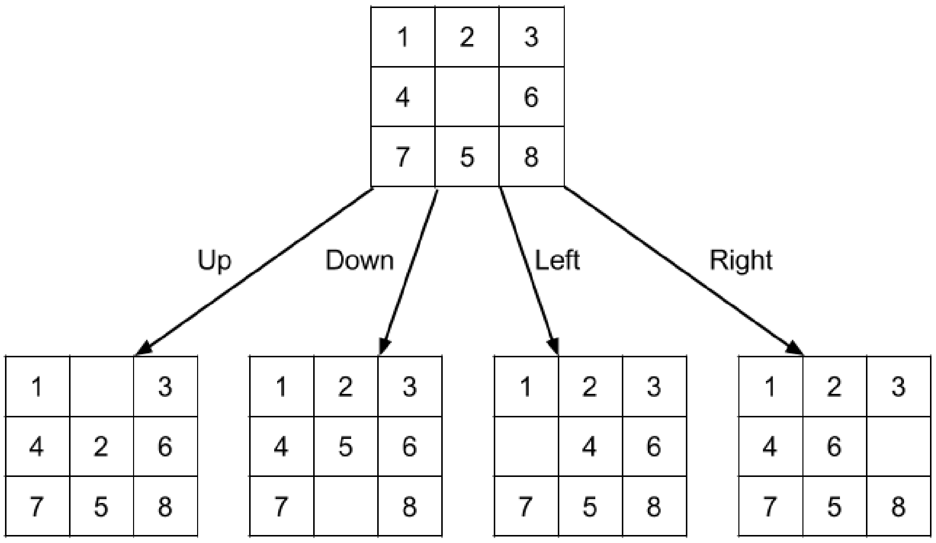

Figure 2.1: An example of the root of a search tree for attempting to solve the 8-puzzle. We observe a current world state and build a computational model of it. We then determine which actions we can take to transform this state and simulate taking each of them to produce a model of the transformed world state.

node to our final state.

While this exhaustive search procedure may work for small problems with a limited number of moves available from each state it becomes infeasible for moderately complex problems. As an exhaustive search considers all possible states without bias the number of nodes which must be considered grows exponentially with the number of moves required for the simplest solution. A common approach to resolve this issue is to attempt to focus the search on the most promising branches, expanding nodes which we believe are 'close' to a goal state. This is typically done by using a heuristic function which gives an estimated fitness value for the current node, with nodes which we believe are 'closer' to a goal state having higher fitness values than those that are further away. By choosing to expand the node with the highest fitness we hope to ignore unnecessary branches of the search tree, turning our breadth-first search into a best-first search (Russell & Norvig, 2013).

While heuristic based best first search methods can be very powerful they also come with the non-trivial task of defining a suitable heuristic to produce a numeric value from the world state which can be used to rank the currently known states. While this may be a relatively simple task for some games it is often not possible to define an accurate heuristic function for a state, especially in multi-player games. With this in mind Monte-Carlo based methods have gained traction recently to estimate the value of a particular state by simulating random playouts from tree nodes to terminal states or until some arbitrarily defined horizon is reached.

This is the basis behind Monte-Carlo Tree Search (Chaslot et al., 2008). We begin by picking the tree node with the highest heuristic score which is calculated from success rates of simulations. We then add a new child by simulating a playout which starts with the action we believe to be the most promising, breaking ties randomly. When we have reached a terminal state, which may or may not be a goal state, we increment the number of simulations and successful playouts of the child node we just added, and then backpropagate the playout result up the tree, updating the statistics of each of the new node's ancestors.

The most commonly used implementation of MCTS uses a multi-armed bandit approach to choosing actions at each stage using a heuristic called Upper Confidence Bounds for Trees (UCT) (Kocsis & Szepesvári, 2006), based on the UCB1 algorithm (Auer et al., 2002). This approach uses the heuristic

$$UCT(i) = \frac{s_i}{n_i} + C\sqrt{\frac{\ln N}{n_i}}$$

where s_i represents the number of successful playouts which have involved child node i, n_i is the number of times child node i has been visited, N is the number of times the current node has been visited and C is a constant balance factor. The left hand term of this heuristic represents the expected success rate of the current node while the right hand term is representative of the amount we have explored this node relative to others. The value of C can adjust the balance between exploration and exploitation to make the most efficient use of the resources we have available and is often determined empirically to best fit the problem domain. When choosing which node to expand next we recursively take the branch which leads to the child node with the highest UCT score until we reach a node that we wish to expand.

Note that at any expansion phase of MCTS we only add 1 new persistent node to our tree at a time, any nodes generated during playouts beyond this first node are discarded and used entirely for payoff estimation. This approach means that resource usage is kept low in less promising areas of the tree and is used instead for areas of the tree where we are more confident that our solution exists, dedicating resources to where they are most effectively used and making MCTS a very efficient and attractive option for low powered systems, or for situations when the world state is complex.

While MCTS is typically used for action selection in problems which are discrete in both state and action space techniques exist to allow its use in continuous scenarios (Couëtoux et al., 2011).

2.2.2 Prediction

For any form of deliberative planning we are required to generate new states based on an assumed world state and some sequence of actions. In order to do this we propose the use of an on-board physics simulator which the agent can initialise with the world state, apply actions, and observe predicted results without actually modifying its environment.

Simulation of physical processes and expected behaviours has been utilised widely in the robotics community and in combination with the field of artificial life. The most obvious use of real world simulation in robotics is in performing experiments to test new control algorithms without the expense, risk and time associated with deploying on physical robots, costs which grow rapidly when we extend from single agents to swarm robotics.

A number of robot specific simulators have been developed such as V-REP (Rohmer et al., 2013), Player/Stage (Gerkey et al., 2003) and Webots (Michel, 2004) which all have their uses, however each of them have their own individual flaws and, perhaps most importantly here, tend to struggle to simulate more than a hand full of entities at reasonable speeds.

The ARGoS simulator (Pinciroli et al., 2012) counters this as it is designed from the ground up to be used for swarm robotics and has an architecture which enables dozens of robots to be simulated faster than real time.

Similarly to the work presented in (Winfield et al., 2014; Vaughan & Zuluaga, 2006) the use of a simulator on-board each agent to predict the utility of available actions can be used to allow agents to explore novel solutions without physically interacting with the real world. In this work we are interested in the use of physics simulations rather than simulations of other agents as explored in (Weitnauer et al., 2010). The use of physics simulators for prediction within robot controllers is gaining popularity as can be seen in (Ornan & Degani, 2013; Kroger et al., 2008), and investigation into improving the reliability of simulations of uncertain terrains is underway (Ferworn et al., 2013). The use of an on-board simulator to explore actions as opposed to controllers and morphologies allows for far faster, cheaper, and safer exploration of possibilities. The use of simulation for AI is widely practised but has only recently made its way into robotics as CPUs have become small enough to be mobile whilst also being powerful enough to run simulations faster than real time.

Due to the nature of these simulations it is possible to predict the change of both static and dynamic environments. While the work presented here will only consider the application of simulations to environments in which objects do not move independently the techniques developed could be used in dynamic environments, subject to increased uncertainty. Given a model of the behaviour of other autonomous agents in the environment, such as humans or other robots, it is possible at each time step of the simulation to predict the next action of the independent agents and simulate their movements, and how those movements could impact the environment the robot is required to operate in. This technique of predicting the behviour of other agents in onboard simulations of dynamic environments has gained a great deal of interest in recent years, and is a key part of some seminal works on Ethical Robots (Winfield et al., 2014).

2.3 Dimensionality Reduction

In this section we provide a brief overview of the concept of associative memories, followed by a Natural Language Processing technique which inspired our means of recalling individual objects. We then provide an overview of each of the dimensionality reduction techniques we have analysed: Principal Components Analysis (PCA), Sammon mapping, Isomap, Landmark Isomap, Local Linear Embedding (LLE), Laplacian Eigenmaps, Kernel PCA, SNE, SymSNE and tSNE. The key features of how each technique works are provided.

2.3.1 Associative Memory

Typical memory systems rely on address based lookup of data which has previously been stored. While this can be a useful mechanism in many cases it has limited utility in the field of cognitive robotics and AI. A more typical requirement in robotics and AI is to be able to recall a piece of information based on some observation or concept. This style of memory is known as either an associative memory or equivalently as content-addressable memory.

Many of these techniques rely on neural networks to perform most of the computation. Neural networks provide a computationally efficient way of automatically learning relevant data representations which are tied to the underlying computational methods of the network architecture.

While many approaches have been investigated, perhaps most influentially the Hopfield network (Hopfield, 1982) for example, the vast majority of approaches rely on a fixed size pattern. This limitation provides two significant hurdles to our system by viewing memories at varying levels of granularity. We first address scenes with varying numbers of objects then move on to the storage of individual objects and the reuse of memories which contain similar objects.

Firstly, the number of objects in a given scene will change between observations. This is further complicated by the possibility of the most appropriate memory containing a different number of objects to the current problem, for example an object may be present in the current scene which will not impact the optimal solution. Since our scene and memories (referred to as keys from here) will require a representation of the objects in the scene and their relative locations, the size of the key will naturally be correlated to the number of objects represented by the key. This issue is one which has been studied but no suitable solution has yet been found which would lend itself to our system, although the concept of holographic reduced representations proposed by Tony Plate (Plate, 2003) may provide useful inspiration. The direct application of holographic reduced representations would likely not be useful in our case, however, as the addition of new items tends to result in large shifts in the combined representation, reducing the likelihood of a success match for similar scenes with different numbers of objects.

Secondly, the meshes associated with learned objects and used in physics simulations will naturally have a varying number of vertices. This would further increase variation in the size of the key and would provide further difficulty if the most appropriate memory to be recalled contains objects which are distinct but respond to physical interaction similarly. To resolve this issue would require a fixed length representation which is also able to convey similarity between objects.

These problems both resemble issues which are commonly faced in natural language processing, namely recall of documents and words which have similar semantic meaning. As such we intend to investigate the potential to apply natural language processing recall techniques to recall of physical objects.

In the document recall case a database is queried with a sample body of text which could range from a single sentence to a multi-page document with the expectation that documents which are based on similar subjects or have similar content should be returned.

Commonly a “bag-of-words” approach is used in these scenarios where an input vector consists of, potentially, thousands of values with each entry representing the frequency of a single word in the system’s vocabulary. These vectors are usually very sparse resulting in a large amount of wasted memory and computation during the matching process. This approach would also fail to recall documents where the wording used varies slightly such as “I walked down the road" and “We went along a street" as similarity between words is not considered.

To address the problem of allowing the meaning of words to be compared the concept of word embeddings has been proposed (Mikolov et al., 2013). A trained word embedding provides a vector representation for each word based on the context the word is often found in during training. For instance the words “man” and “woman” will often be surrounded by similar words and so will be placed in similar areas in space. A more interesting property however is that semantic relationships are also preserved owing to relative consistency in the meanings of words as they are changed between forms. This uniformity in context change means that $v(run) - v(ran) \approx v(pick) - v(picked)$ and $v(king) - v(man) \approx v(queen) - v(woman)$.

An analogous situation with object embeddings might be to transfer the expected behaviour of a meterial between objects of differing shapes, allowing for inference of the semantics of an object or its properties in the environment. Let us consider, for example, that an agent has previously encountered a wooden sphere, and a rubber sphere. From this it maybe be possible to extract a vector which represents the difference between wooden and rubber objects $v_{wr} \approx v_r(sphere) - v_w(sphere)$. Now, if our agent encounters a rubber box for the first time, it may be possible to predict how it will react if existing knowledge of a wooden box exists, predicting as $v_r(box) \approx v_w(box) + v_{wr}$, in much the same way as semantic meaning can be transfered between words in the NLP case. While we do not explore this possibility in this work we believe that the potential to extend this work to explore the possibility of semantic transfer of object properties is very powerful, and could provide increased confidence in agents being deployed in novel environments.

In the simplest case this type of embedding for physical objects could allow our agents to reuse solutions to problems which include similar objects to those in the current scene. It is also possible, however, that correlations may exist between object differences in embedding space and actions required in the solution space. If these correlations do exist the ability for agents to modify recalled solutions based on the discrepancy between keys could provide a significant speed up in the search process.

In a number of NLP techniques mathematical functions are used to combine multiple word representations into a new vector. In these cases the function being used is often context dependent and relies on certain structures in language, requiring the query being processed to be parsed first so that the appropriate functions can be applied in the correct order. One solution which is robust to these shortfalls is presented in (Le & Mikolov, 2014), in which a technique is presented which learns vector representations of paragraphs by aggregating individual word vectors.

In our case, however, there are no relationships between the objects in a scene, and as such this method of combining individual object representations becomes infeasible. In situations where objects can be found to be interacting in certain ways, for example if objects are stacked or resting against each other, then similar techniques may prove to be useful, however we do not consider them in this work.

Although using natural language processing techniques seems the most intuitive way to combine representations of individual objects owing to its inspiration of the underlying representation the problem of matching collections of oriented feature vectors is one which is more closely related to the loop closure/kidnapped robot problems from SLAM and to sparse correspondence of point cloud data. As such computer vision techniques may prove to be a more effective field to take influence from.

2.3.2 Principal Components Analysis (PCA)

PCA (Hotelling, 1933) finds a linear subspace in which the principal axes are aligned along the directions of highest variance in the data to be reduced. By discarding the dimensions of this subspace which represent the lowest variance we can reduce the dimensionality of the space while retaining as much information as possible.

PCA is a statistical analysis technique which maximises the amount of information retained about a distribution of points. The data is first adjusted to have a mean of 0 by subtracting the mean from each of the points in the distribution. We the calculate the covariance matrix of this data, before performing Eigendecomposition on the covariance matrix.

This Eigendecomposition provides us with a set of orthogonal axes and a set of Eigen values which can be used to determine which of the axes represent the directions of highest variance in the data. From these values we can calculate what proportion of the variance is captured by each axis, and discard those which provide little to no value. The axes which represent only negligible amounts of captured variance can often be attributed to noise in the data and so can be removed to not only reduce the amount of data needed to be stored about each point, but also to help generalisability of the captured model.

2.3.3 Sammon Mapping

One of the weaknesses of PCA is that it gives little focus to preserving small pairwise distances which help define the structure of the data cloud. To combat this *Sammon mapping* (Sammon, 1969) produces a similar embedding to PCA while giving higher precedence to small pairwise distances. In both PCA and Sammon mapping we minimise a cost function which contains a sum of functions of pairwise distances. Sammon mapping scales each term in this summation by the inverse of the initial pairwise distance.

2.3.4 Isomap

Isomap (Tenenbaum et al., 2000) aims solely to retain pairwise distances of points which lie on a manifold in the original space. Isomap creates a graph of nearest neighbours and uses the pairwise geodesic distance across this graph as target distances to retain in the low dimensional embedding. This has the effect that curved manifolds can be “flattened” by the embedding, however it is also possible for sections of the manifold to be connected incorrectly resulting in many geodesic distances being underestimated and, ultimately, a poor embedding.

A common example of when Isomap produces high quality results is when applied to a “Swiss roll” dataset. In this case the X and Y components of data points can be approximated by a spiral, and the Z component is randomly distributed.

Since the data effectively lies on a plane which has been rolled up, Isomap can “unroll” the data, forming a 2D distribution of points from the original 3D point cloud. This can both help to reduce the amount of data stored, and help various machine learning techniques to process the data more accurately, reducing the possibility of relationships being inferred as a result of the extra dimensions in the original data.

2.3.5 Landmark Isomap

Landmark Isomap (Silva & Tenenbaum, 2003) aims to reduce the computational complexity of standard Isomap by performing the most expensive operations on a subset of the data. A subset of data points, referred to as landmarks, are selected and Multi-Dimensional Scaling (MDS)

(Torgerson, 1952) is applied to produce a low dimensional embedding of the landmarks. The distances of each point to these landmarks is then used to produce a low dimensional embedding of the remaining points.

2.3.6 LLE

Local Linear Embedding (LLE) (Roweis & Saul, 2000) calculates the position of each point in the high dimensional space as a linear combination of its nearest neighbours, working under the assumption that the manifold on which the data lies is locally linear. A low dimensional embedding is then found which aims to allow each of the embedded points to be reconstructed from a linear combination of its neighbours using the same weights as were used in the higher dimensional space.

2.3.7 Laplacian Eigenmaps

Laplacian eigenmaps (Belkin & Niyogi, 2003) attempt to maintain the distance between each point and its nearest neighbours by producing a nearest neighbour graph before minimising a cost function in which the distance of each edge to a neighbour in the low dimensional embedding is weighted based on the same distance in high dimensional space. This can be defined as an eigenproblem and solved appropriately.

2.3.8 Kernel PCA

Kernel PCA (Schölkopf et al., 1998) is, as its name suggests, a variant of PCA in which a kernel is first applied before eigendecomposition is performed on the kernel matrix as opposed to the covariance matrix as in PCA. It is the application of eigendecomposition in kernel space which allows kernel PCA to perform non-linear mappings.

2.3.9 SNE

Stochastic Neighbor Embedding (SNE) (Hinton & Roweis, 2003) aims to preserve the probability that a point will 'select' another point as its neighbour. The probabilities in the high dimensional space are calculated using the similarity and distribution of points before a low dimensional embedding is calculated which aims to preserve these by minimising the sum of Kullback-Leibler divergences between the high and low dimensional distributions over nearest neighbours.

2.3.10 SymSNE

SymSNE (Cook et al., 2007) is very similar to standard SNE with the exception that pairwise similarities are assumed to be symmetric and are calculated as the mean of the pairwise similarities calculated in standard SNE.

2.3.11 tSNE

tSNE (van der Maaten & Hinton, 2008) builds on SNE by using the same symmetrical similarity values used in SymSNE with the addition of using a student's t-test in the low dimensional space rather that a Gaussian as is typically used in SNE/SymSNE. This allows some of the shortcomings of SNE/SymSNE to be overcome when aiming to optimise the cost function of the low dimensional space.

Using the techniques presented here we will generate a number of embeddings from raw observation data which can then be used to recall objects which have moved in similar ways as a result of robot interactions. This will then allow scenes to be recalled which do not contain the same objects but which do contain objects which we would expect to react in similar ways when various actions are performed.

2.4 Scene Recall

He we provide an overview of a number of the techniques which could be used in order to recall collections of objects, independent of the number of objects in the scene. We begin by exploring a bio-inspired approach based on saccadic eye movements and leaky integrate-and-fire neuron models. We then look at computer vision techniques which are used to identify familiar scenes during loop closure, a common problem in SLAM.

2.4.1 Human Vision

Saccadic eye movements are a natural phenomenon when observing and processing visual scenes. When processing a scene the eyes move between various areas of interest in order to gather information about the environment. An article from 1997 (Gilchrist et al., 1997) identified an individual without the ability to move their eyes and compensated for this by performing saccadic style movements with their head when performing visual tasks, with little discernible visual impairment. This suggests that saccadic eye movements form a core part of our ability to process the environment.

The Leaky Integrate-and-Fire (LIF) model of neurons is a simple model of a neuron which models the electrical properties of a biological neuron as a simple resistor-capacitor model. A brief summary of the history the LIF model can be found in (Brunel & van Rossum, 2007).

Generally, a LIF neuron can be modelled as a capacitor, representing the held charge in a neuron, and a discharge resistor across the terminals of the capacitor, representing the leakage of ions through the cell membrane, reducing the charge in the capacitor. Once the neuron reaches a sufficient level of charge it produces an electrical spike, reducing its charge back to some base level. As incoming spikes from other neurons arrive the level or charge increases, eventually causing an output spike if the incoming spikes arrive at a sufficient rate.

2.4.2 Computer Vision

A number of techniques exist for analysing scenes in the field of computer vision. In many of these techniques the analysis takes place in two parts. The first is a static analysis of a single video frame, extracting features from a raw image feed. The second stage is to correlate these features between frames, inferring information about how objects in the scene have changed, or how the camera perspective changes over time.

In many cases, for instance in SLAM scenarios, it can be assumed that the scene will not change much between frames, meaning that the location of detected features should not change much between frames. In these cases the proximity of related features between frames can be exploited to perform more computationally efficient comparisons. In both loop closure and the kidnapped robot problem, however, we do not have this guarantee of proximity.

In the case of loop closure we are attempting to detect if the scene we are observing matches part of a generated map which we have already visited before, enabling us to “close the loop” on the map. Consider the case of a circular corridor for example. A robot could begin its mapping process using SLAM, only to reach its starting point as part of this process. While a combination of dead-reckoning and movement estimation from vision aim to minimise positional error as part of the mapping process they are both noisy approximation methods of positioning. They also provide relative movement information, meaning that errors gradually accumulate over the course of the mapping process. As such we may have an estimate of which part of the map we are currently close to, but it is common for the loop closure process to assume no knowledge of location. (Eliazar & Parr, 2004) provides further discussion of this problem, along with a SLAM algorithm which addresses the problem of loop closure, and a survey of loop closure techniques can be found in (Lowry et al., 2016).

Similarly, the kidnapped robot problem addresses the problem of a robot which has a map of various locations which is then moved to an unknown location, having no visual input during the moving process. The problem is then to determine which part of the map it is in when visual input returns. This is a very similar problem to loop-closure, with the guarantee that no location information can be used.

2.5 Summary

In this chapter we have presented an overview of the literature surrounding the ideas presented in chapter 3 and provided background for the work to be presented in chapter 4. We recap the relevant content in chapter 4. In the following chapter we describe the production of a dataset which can be used for the creation of object embedding spaces, and present our work on it.

Chapter 3

Proposed System & Planning Protocol

Here we detail the scenario which motivated this work and provide a high level architecture for our proposed planning system. We provide context for the work presented later in this thesis and justify our decisions to make use of the techniques used when exploring our hypothesis.

3.1 Problem Scenario

Consider a search and rescue situation in which a swarm of robots has been deployed in order to find survivors in an environment which would prove dangerous for a human search and rescue team. We are specifically concerned with cases where these search and rescue operations take place in man-made areas, where it is likely that identifiable objects will be present.

During the search process, the swarm is performing Simultaneous Localisation and Mapping (SLAM) and is guided by a frontier-based approach to help ensure that the entire area is explored without prior knowledge of the area. A new frontier to explore is discovered, however access is blocked by a collection of objects. In this work we assume that these objects can be identified individually using object-based SLAM techniques, such as SLAM++.

To enable further exploration the swarm must now develop a sequence of actions that can be carried out by a subset of the robots present in order to clear the obstruction, and allow exploration to continue. As the objects can be identified a physics simulation of the obstruction can be initialised, allowing robots to predict the result of various interactions with the detected objects.

With this knowledge it is possible for individual robots to begin executing planning procedures. As the solution space is infinite it is important for robots to be able to search the space efficiently, making naive search processes unsuitable for this application. Due to its inherant balancing between exploration and exploitation we propose the use of Upper Conifdence Bounds for Trees (UCT) for a best-first tree search as a planning algorithm. Similar search techniques have already been shown to have applications for real-time action planning scenarios in the physical travelling sales person problem. Best first seach techniques, specifically Monte-Carlo Tree Search, have also been shown to be a suitable candidate to be augmented with a memory system to help with solving complex problems.

Without generalisability, however, there would be no benefit to adding a memory system as, given the number of possible combinations of items and relative locations, it is highly unlikely that matching memories would be found. As such, a fuzzy recall method is required for the scene recall process. We begin by generalising the recall process for individual objects, before extending to multi-object scenarios. In the case of single object recall we are interested entirely in the way in which objects behave when they are interacted with, rather than based on visual information or shape data.

We need only recognise that two books may be visually distinct but behave in very similar ways when pushed to see that this is the case. Conversely, two equally sized boxes may look visually similar, and have similar shape and size, but could respond very differently to being interacted with if their densities are different, or if one has a higher coefficient of friction than the other.

It could also be the case that objects which do not look alike, and which have different shapes, could behave in very similar ways to interactions. Consider the case of a book with an overhanging cover and a small box for instance; the overhanging cover of the book would cause difficulties for shape matching algorithms, and the visual difference would result in poor visual matching. Despite this, however, it is quite possible that they can be manipulated in the same way to produce the same effect.

We propose the use of an Embedding Space to assist with this problem. Based on the idea of neural word embedding spaces, spaces in which points which represent embedded items with similar semantic meaning, we aim to develop a space in which objects which behave in a similar manner to one another are in a similar region of the generated embedding space. This can be seen as an unsupervised training technique, in which a database of objects to be stored are presented and, by interacting with those objects, an agent can observe the way an object responds to being interacted with in various ways. This will produce a large amount of observation data for each object presented, which can then be used to create an embedding space. This space is efficient to search, having captured the dynamic behaviour of the objects in a relatively low dimensional space compared to raw observation of the objects, and has the useful property that, by construction, objects close to a “query” object aught to behave in a similar manner to it.

Given that the generation of this embedding space reduces the dimensionality of the object space compared to the raw data obtained by observing how the object responds to being interacted with we refer to points in it as “reduced response vectors” (RRVs). When executing a recall process we can take the RRV of an object identified in the scene and use it to query an embedding space which contains all known objects. We know that any RRVs which are a close match to the query object will behave in a similar way to the query object when interacted with. As such, given that in all cases we are aiming to clear an obstruction, any information that we have about moving an obstruction involving the recalled object is likely to be pertinent to solving the same problem containing the query object.

With a means of capturing the similarity in behaviour of individual objects it is necessary to expand this recall process to collections of objects in certain configurations, referred to as “scenes”. Clearly, it is important to capture the location of objects in a scene for meaningful recall to take place. The lack of a global coordinate frame means that we can consider only relative positions of objects. With few guarantees on the structure of each scene it is not possible to simply declare an object as the centre of a scene and use that as the basis of a coordinate frame for all objects present. As such we consider the relative position between all pairs of objects when defining a scene. Formally, given a set of objects O, in a scene S, with pose $p(o)$, a scene is defined as:

$$S = \{\forall a \in O, \forall b \in O \setminus \{a\} | (a, b, p(a) - p(b))\} \tag{3.1}$$

It is important to note that, with this format, scenes are likely to contain differing numbers of objects, resulting in different sized scene definitions. This introduces complexities when recalling scenes as the most relevant memory is not guaranteed to have the same number of objects as the query scene. Consider a query scene which contains an object which will not be involved in the manipulation of other objects to clear an obstruction.

At the start of the search process there is no way of filtering this object from the query scene as it is not known that the object will not play a part in later planning stages. It could be that the most relevant memory is an almost identical scenario, differing only by the lack of the uninvolved object. For situations such as this we must make use of recall techniques which are agnostic to the exact number of items in a scene. We propose two possible solutions to this problem.

The first takes inspiration from SLAM systems, specifically in the need for loop closure - the ability of a robot to recognise when it has returned to a previously visited location in order to “close the loop” in its internal map.

The other is a bio-inspired technique based on saccadic eye movements - the movement of the focus of the eyes between areas of high information density when observing a scene. By combining this with a leaky integrator style system, often used in spiking neural networks, we can model each scene as a single leaky integrator. This allows us to declare a matching recall in the case that an integrator passes a threshold, in effect causing the “neuron” to fire, or to simply take the memory with the highest potential when some recall limit has been reached, be that time based or otherwise.

With the ability to recall scenes, the search process also needs a way to check if a goal state has been reached. In our case, a goal state is any state in which a path is available from all robots' locations to the opposite side of the obstruction. A standard path planning algorithm can be used for this, with a goal state being one in which the path planning algorithm succeeds.

Path planning is a problem which has many possible solutions and which has been studied extensively in computer science, and in robotics. As such, we do not consider any path planning work here. For efficiency, we propose that a 2D path planning process be used, with a 2D map being generated by simply discarding the Z component of all vertices in an objects mesh to produce an X, Y projection of the objects in the scene.

The planning process we propose is based on best-first tree seach techniques, modified to be more applicable to our problem domain. Often, best-first search is used in AI applications in which a sequence of moves take place, with moves being performed by multiple "players". In these cases the order in which moves are played can imply information about opposing players. As a result of this search nodes representing the same world state, reached through different action sequences, are considered as distinct elements of the search tree.

In the case of a single agent, however, this is not the case. If a state is reached as a result of two separate sequences of actions then there is no benefit to having them both in the search tree. Only the sequence of actions which uses the fewest resources need be considered. Any future nodes reachable from one node (A) are guaranteed to be reachable from another node (B) as long as the world state in B is the same as that in A, and as long as the amount of resources available at node B is greater than or equal to the resources available at node A.

We therefore propose the following modifications to a simple best-first tree search. When building the search tree we build up a map from encountered world states to search nodes. When a new node is added to the tree we first check to see if the world state is already attributed to a search node. If not then we simply add the new node as usual, updating the state to node map in the process.

If the new node represents an already encountered world state, however, then we need to ensure that only the more optimal route to that world state is considered. We do so by comparing the resources available in the existing node (E) with the new node (N). If E has more resources available then we simply mark the action that generated N as invalid so that it will not be expanded in future, and we discard the new node.

If N has more resources available, however, we must transfer the descendent nodes of E to N. As the world states are the same, and the actions available from E are guaranteed to be a subset of those from N, we can simply transfer each subtree leading from an action in N to the equivalent action in E. The available resources in the descendants of N must now be updated. Once this process is completed we can mark the action that generated E as invalid. Finally, we back propagate from N and E to reflect the new location of the subtree, before updating the state to node map.

As this modification means that multiple routes can now lead to the same search node we ostensibly convert a tree search in to a graph search process, where branches are pruned and transplanted to other sections of the tree as appropriate. As such we refer to this process as Transplanted Tree Search (TTS).

Best first searches (and therefore TTS) require a utility function which determines the anticipated quality of a given state. When applying UCT this state score is then used to balance exploration and exploitation in order to effectively cover the search space. In our case we propose that the smallest gap along any path found from the current position of the robot to the target location is used as a state utility function. A score of $\frac{width_{min}}{width_{robot}}$ is given to a state, where $width_{min}$ represents the width of the smallest gap along a discovered path, or 0 when no path is found, and $width_{robot}$ represents the width of the robot when travelling forwards. By using this scoring mechanism a score of 0 is awarded to a state when no path is available, and a score of ≥ 1 is given to a state in which a valid path is found.

While a simple binary score of 0 or 1, depending on the availability of a path which the robot can traverse, could also be used it would not bias the search in the early stages. A binary score would only provide any feedback when a valid solution is found, leaving all of the biasing in the search process to the memory system.

In order to combine TTS and our proposed recall system to produce an experience based version of TTS (which we refer to as EBTTS) we simply add a third term to the UCT function. The standard UCT function is defined as

$$UCT(i) = \frac{s_i}{n_i} + C\sqrt{\frac{\ln N}{n_i}}$$

where i represents a given move, s_i is the expected "score" of the state which i leads to, n_i is the number of times we have explored the action, and N is the number of times we have visited this node.

The first term in the UCT function exploits our existing belief about the state of the world after we perform an action. The second term is used to bias the search to underexplored areas of the search space. We can add a further bias to this function by simply recalling our knowledge of previously encountered problems.

If we have previously encountered a similar problem then we already have some knowledge of the utility of various actions in the previous problem. If the previous problem and the current one are similar enough then we can reuse this knowledge to bias us toward actions in the current problem which are more likely to lead toward a goal state. If we have encountered the exact same problem before then we should give a higher priority to our existing knowledge than if we have only encountered a slightly similar problem. As such, the experience based UCT function (EBUCT) should takes the form

$$UCT(i) = \frac{s_i}{n_i} + C\sqrt{\frac{\ln N}{n_i}} + R\, q(s,m)\; e(i,m)$$

where R is a constant factor, used to balance the influence of recall, $q(s, m)$ is the "quality" or similarity of the most appropriate memory (m) relative to the current scene (s), and $e(i, m)$ is the expectated score of move i as stored in memory m.

Although not explored in this work it is expected that new memories would be retained when a scene is encountered during the search process which is sufficiently different from any other scenes currently in the agent's memory. There a numerous ways which could be used to determine when a sufficiently unique scene is encountered, however the analysis and development of such techniques is beyond the scope of this thesis.

Chapter 4

Object Recall

In this chapter we address the challenge of allowing an agent to recall objects which they have experience of interacting with in a way which allows knowledge transfer between these objects. As noted previously, for the problem domain we are concerned with, we are concerned with the ability to recall objects which behave in similar ways when interacted with by the agent.

We begin by introducing the motivation for the work in this chapter, before briefly recapping the dimensionality reduction techniques presented in chapter 2. Then, we describe the benchmark dataset which was created to facilitate this work and which is available for future work. We then present the experimental methodology used to compare the generated embeddings. Finally, we summarise our findings in this chapter.

The work presented in this chapter has previously been published in (Redpath et al., 2017).

4.1 Introduction

Robots often operate in structured environments when deployed for practical applications, with those applications ranging from domestic assistance robots to search and rescue. As these environments are predominantly man-made it is not uncommon for robots to operate in environments in which they are surrounded by repeated, identifiable objects. Furthermore, these objects are likely to be arranged in similar configurations to one another, for instance a chair tucked under a table, or a book on a shelf. It therefore seems beneficial for robots to be able to make use of these repeating objects and structures in order to help accelerate action planning when new problems are encountered, if they have already solved a similar problem in a similar environment.

In order to make use of these existing solutions the robot must be able to recall a collection of objects in a specific configuration, known here as a "scene". For our purposes a scene can be fully defined by a set of objects, and their locations relative to one another. When attempting to recall these scenes, however, it is beneficial to be able to transfer solutions to guide searches between scenes which not only contain the exact same objects, but also objects which react in similar ways. A simple example of this could be a book and a box of similar size. The objects would be identified as different types of object, however any solutions which contain a book are likely to be beneficial to problems containing a box in a similar location. As such, it is important that objects can be recalled based on the way they react to the robot's actions, rather than how they look or what their shape is explicitly.

With the requirement of recalling objects based on the way they respond to interaction in mind we now aim to find a computationally efficient way in which to quantifiably compare two objects in order to enable recall of complete scenes. The requirement of quantifiable similarity is derived from a desire to adjust the impact recall has on biasing a search process based on scene similarity. When similar scenes are found in memory they should have a greater impact on the search process than scenes which have very poor similarity.

To this end we explore the application of a number of dimensionality reduction techniques to raw observations of object movement in order to produce a lower dimensionality representation of each object. This lower dimensionality representation can then be used to compare objects in a generic manner which will allow extension of individual object recall to various forms of scene recall. To the best of our knowledge, this is the first time work has been carried out to explore the ability to recall objects based on how they react to robot manipulation.

As a result, this work gives rise to two key contributions. The first is a benchmark dataset which can be used for any future efforts to explore this problem. The second is the exploration of various dimensionality reduction techniques.

4.2 Related Work

In this section we provide a brief overview of each of the dimensionality reduction techniques we have analysed: Principal Components Analysis (PCA), Sammon mapping, Isomap, Landmark Isomap, Local Linear Embedding (LLE), Laplacian Eigenmaps, Kernel PCA, SNE, SymSNE and tSNE. The key features of how each technique works are provided.

PCA (Hotelling, 1933) finds a linear subspace in which the principal axes are aligned along the directions of highest variance in the data to be reduced. By discarding the dimensions of this subspace which represent the lowest variance we can reduce the dimensionality of the space while retaining as much information as possible.

One of the weaknesses of PCA is that it gives little focus to preserving small pairwise distances which help define the structure of the data cloud. To combat this *Sammon mapping* (Sammon, 1969) produces a similar embedding to PCA however gives higher precedence to small pairwise distances. In both PCA and Sammon mapping we aim to minimise a cost function which contains a sum of functions of pairwise distances. Sammon mapping scales each term in this summation by the inverse of the initial pairwise distance.

Isomap (Tenenbaum et al., 2000) aims to retain pairwise distances of points which lie on a manifold in the original space. Isomap creates a graph of nearest neighbours and uses the pairwise geodesic distance across this graph as target distances to retain in the low dimensional embedding. This has the effect that curved manifolds can be "flattened" by the embedding, however it is also possible for sections of the manifold to be connected incorrectly resulting in many geodesic distances being underestimated and, ultimately, a poor embedding.

Landmark Isomap (Silva & Tenenbaum, 2003) aims to reduce the computational complexity of standard Isomap by performing the most expensive operations on a subset of the data. A subset of datapoints, referred to as landmarks, are selected and MDS is applied to produce a low dimensional embedding of the landmarks. The distances of each point to these landmarks is then used to produce a low dimensional embedding of the remaining points.

Local Linear Embedding (LLE) (Roweis & Saul, 2000) calculates the position of each point in the high dimensional space as a linear combination of its nearest neighbours, working under the assumption that the manifold on which the data lies is locally linear. A low dimensional embedding is then found which aims to allow each of the embedded points to be reconstructed from a linear combination of its neighbours using the same weights as were used in the higher dimensional space.

Laplacian eigenmaps (Belkin & Niyogi, 2003) attempt to maintain the distance between each point and its nearest neighbours by producing a nearest neighbour graph before minimising a cost function in which the distance of each edge to a neighbour in the low dimensional embedding is weighted based on the same distance in high dimensional space. This can be defined as an eigenproblem and solved appropriately.

Kernel PCA (Schölkopf et al., 1998) is, as its name suggests, a variant of PCA in which a kernel is first applied before eigendecomposition is performed on the kernel matrix as opposed to the covariance matrix as in PCA. It is the application of eigendecomposition in kernel space which allows kernel PCA to perform non-linear mappings.

Stochastic Neighbor Embedding (SNE) (Hinton & Roweis, 2003) aims to preserve the probability that a point will 'select' another point as its neighbour. The probabilities in the high dimensional space are calculated using the similarity and distribution of points before a low dimensional embedding is calculated which aims to preserve these by minimising the sum of Kullback-Leibler divergences between the high and low dimensional distributions over nearest neighbours.

SymSNE (Cook et al., 2007) is very similar to standard SNE with the exception that pairwise similarities are assumed to be symmetric and are calculated as the mean of the pairwise similarities calculated in standard SNE.

tSNE (van der Maaten & Hinton, 2008) builds on SNE by using the same symmetrical similarity values used in SymSNE with the addition of using a student's t-test in the low dimensional space rather that a Gaussian as is typically used in SNE/SymSNE. This allows some of the shortcomings of SNE/SymSNE to be overcome when aiming to optimise the cost function of the low dimensional space.

4.3 Description of Dataset Generation and Format

Here we describe how the dataset used was generated. We explain the setup of the simulations which were used to create raw movement logs and the format of the data in the published dataset, available at `https://www.richardredpath.co.uk/research.html`.

When carrying out this work we made use of three datasets to evaluate the techniques explored. These datasets are designed to explore the behaviour of the embedding techniques on simple objects with few different classes of object (referred to as the "primitive" set); complex objects which are more representative of items which are likely to be encountered in deployed systems (referred to as the "complex" set); and a combination of these two sets, allowing for analysis of the techniqes on larger datasets which may contain a number of very similar objects along with a number of very specific objects (referred to as the "combined" set).

The "primitive" set is a procedurally generated set of cuboids and spheres of various sizes. As this set only contains two classes of object it has the benefit of allowing visual examination of the results in order to gain greater insight in to the behaviour of any embedding techniques.

The "complex" set conatins meshes which were taken from the KIT object database (Kasper et al., 2012). The KIT object database is a collection of OBJ mesh files of various household objects which have been 3D scanned and so provides a highly detailed representation of a large collection of real world objects. In order to increase computational efficiency and reduce noise in the physics simulations the meshes in this dataset were heavily simplified using a mesh simplification algorithm available at (Forstmann, 2016).

The "combined" set is simply the combination of these two sets.

All of the mesh files used for the creation of this dataset were in OBJ format. Each object is loaded in to a simple scene in the V-REP simulator, which contained only a robot and the object. The robot used is the default ePuck simulation provided with the V-REP simulator. As objects are loaded their mass is determined using a denisty of 200 kg per cubic meter. The object under observation is placed at the centre of the arena, with the robot placed 3m away from the object. The relative position and orientation of the robot and object are detailed below.

When the simulation starts the robot stays still for 5 seconds, which allows time for the object to settle in to a stable position. After 5 seconds the robot begins to move forward at a constant speed with both wheels set to a target velocity of 2 radians per second. As the robot moves forward it contacts the object, pushing it in the arena. In many cases it is likely the both the object and the robot will be impacted by this interaction. The position, orientation, linear velocity, and angular velocity of both the object and robot are logged at every time step, 50ms in these simulations. We also log any times in which the robot and object make contact to allow for more targeted processing.

While the robots used in these experiments do not have visual sensors of their own they are used simply as an example of an agent interacting with its environment. As a SLAM system would be required for any of the planning processes described in this work we assume that such a system is also available for this training phase. It is expected that in any deployed system a SLAM system would be available. There are many forms this system could take, with an overview of some of the more common techniques presented in chapter 2.

To derive the inital location of the robot we consider the bounding box of each object, providing a total of six faces which can be oriented downwards. For each of these six faces there are four faces which are facing “outwards” from the object, giving a total of twenty four faces from which can object can be approached. For each of these faces we position the robot a fixed distance of 3m away from the bounding box and run 5 simulations per face: 1 with the robot aligned with the center of the face, 2 at an offset of +/-20% of the face width, and 2 at an offset of +/-40%. A graphical representation of this can be seen in Figure 4.1. This results in a total of 120 interactions per object.

We capture the observation data for each run in two separate log files. The first is a csv file which conatins the position, orientation, linear velocity and angular velocity of both the object and robot at each time step. The second is a list of time steps during which a collision was registered by the simulation’s physics engine.

Each of these logs is provided in three formats: one which contains the data for only the robot, one for only the object, and one which contains the data for both robot and object.

The logs provide a means of tracking how an object and robot have moved over a 15 second period, during which they interact, from a variety of starting configurations. We extract the position and orientation of both the object and the robot at multiple time steps and use how they both move to produce a low dimensional embedding space with the aim of capturing similarity between how objects move. These logs are released to be used as a benchmark dataset for future attempts at this problem.

4.4 Analysis of Recall Quality for Embeddings and Views

Here we look at the application of the techniques listed earlier, and which are described in greater detail in chapter 2. We begin by defining the term “views” as we use it here and

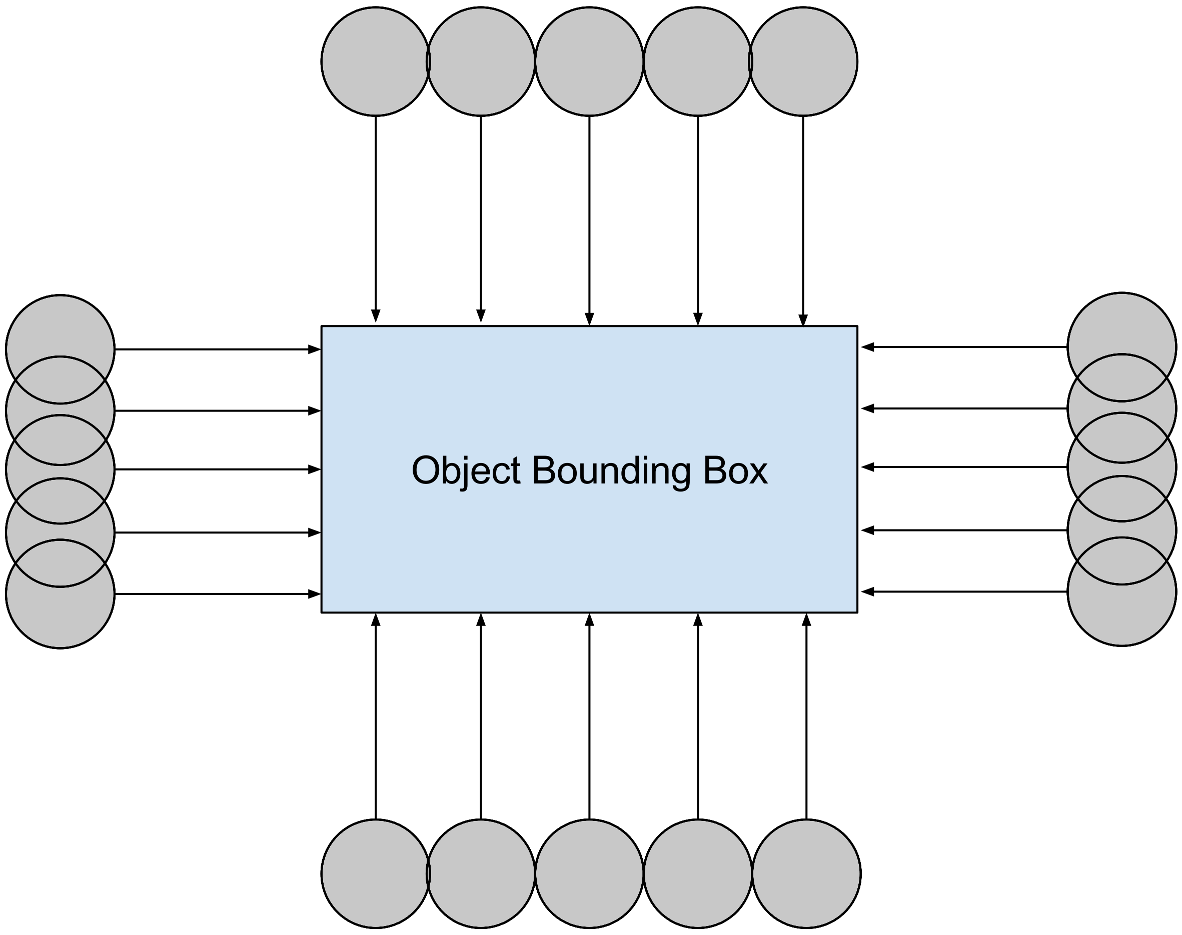

Figure 4.1: Simulation setup used to capture how objects move when interacted with. Robots are initialised to each of the grey circle positions and objects are aligned such that the axes of their bounding boxes align with the axes of the world. After 5 seconds of simulation the robots begin to move forwards. Position, orientation, linear velocity and angular velocity are all logged for both the robot and object every 50ms. The timestamp of the simulation in which contact occurred between the robot and the object are also recorded.

present the views of the data which we explore.

4.4.1 Definition of Data Views

At the time that the work was carried out it was unclear which features of the object and robot motion would be most important to produce a high quality embedding. We therefore looked at various subsets of this data in order to determine which provided the highest quality embeddings. Each of these subsets are referred to as a “view” of the data.

The features collected during simulated interactions were: position(p), orientation in Euler angles (e), orientation as a quaternion (q), linear velocity (l), and angular velocity (a). Clearly, the position and orientation of each object will be important for any scenario in which we wish to manipulate objects in the environment and so they are included in all of the views explored. As two forms of orientation dat were available both were explored. It was unclear if the addition of information regarding the linear and angular velocity of the objects in the scene would be beneficial and so views were explored with and without these features.

This results in a total of eight views of the data, with each view containing positional data, orientation data in either Euler angles or quaternion form, and optionally velocity data.

Each of these views are named based on the letters assigned above. As an example the view which contains positional data, Euler angles, and angular velocity but not linear velocity is referred to as "pea". A summary of which views contain which features is provided in Table 4.1.

View	Position	Euler Angles	Quaternion	Linear Velocity	Angular Velocity
pe	✓	✓			
pea	✓	✓		✓	
pel	✓	✓			✓
pela	✓	✓		✓	✓
pq	✓		✓		
pql	✓		✓	✓	
pqa	✓		✓		✓
pqla	✓		✓	✓	✓

Table 4.1: Features included in each "view" of the data. A checkmark indicates that the feature is present in the appropriate view. All views contain the position of the object and the orientation of the object in either Euler angle or quaternion form. Each view may also contain the linear and angular velocity of the object and robot.

4.5 Experimental Methodology

Here we present the methodology used to extract the data used as the basis for dimensionality reduction. We then present the evaluation metric used to compare the performance of both dimensionality reduction techniques and the views explored.

For each object in a dataset we extract the appropriate features for the view being explored, as defined in Table 4.1. The data from these views is then filtered to only include snapshots of the simulation data at multiple time points for every starting configuration. These extracted features are then stacked to give a single point in "observation space", which is used to represent the object.

We then combine the filtered data for each object to produce a dataset which is used for dimensionality reduction. To each of these sets we apply each of the dimensionality reduction techniques investigated to the entire collection of points, resulting in a low dimensional embedding space which contains a single point for each of the objects we are interested in.

4.6 Evaluation Metric

As the aim of the created embedding spaces is to allow efficient comparison of objects we quantify the performance of each technique as the average difference between an object's coordinate in observation space and the coordinates in observation space of each of its nearest neighbours in the embedding space.

We quantify the error of a particular technique as:

$$E(t) = \sum_{s \in S} \sum_{n \in N(s,t)} \frac{|o(s) - o(n)|}{5}$$

Here, S is the set of shapes the embedding space has been generated from, $N(s,t)$ is the set of 5 nearest neighbours of shape s in the embedding space generated by technique t and $o(s)$ is the position of shape s in observation space, i.e. the vector generated by stacking all observations of the shape s. It was decided to use the 5 nearest neighbours for this dataset in order to extend the neighbourhood outside of near identical objects, but without extending excessively far in to a relatively small database of objects. It is likely that the most appropriate neighbourhood side is dependent on the size of the training set, and the variety of objects in it. Investigating a way to determine the ideal neighbourhood size is beyond the scope of this work.

This error metric is show in Figure 4.2 and Figure 4.3. The points in the upper halves of these images represent the observed behaviour of the objects, with the points in the lower half representing the calculated embedding. In practice the observation points would be in a high dimensional space, with the embedding points in a far lower dimensional space. The light green point represents a single shape s from the equation above, with the darker green points surrounding it representing its neighbours n. The nearest neighbours are taken in the embedding space, and their corresponding points in the observation space are also highlighted. The left hand diagram represents a successful embedding with the nearest neighbours in the low dimensional space also being situated close to the query object in the high dimensional space. The right hand image, however, gives an example of a poorly performing embedding, with the nearest neighbours in the low dimensional space representing objects which do not behave similarly to the query object when viewed in the observation space.

4.7 Results

We now present the results of the techniques explored to the primitive dataset, the complex dataset and the combined dataset. In order to create these embeddings we utilised the Matlab Toolbox for Dimensionality Reduction released along side (van der Maaten et al., 2009). With no information about the spatial distribution of the observation space we apply a range

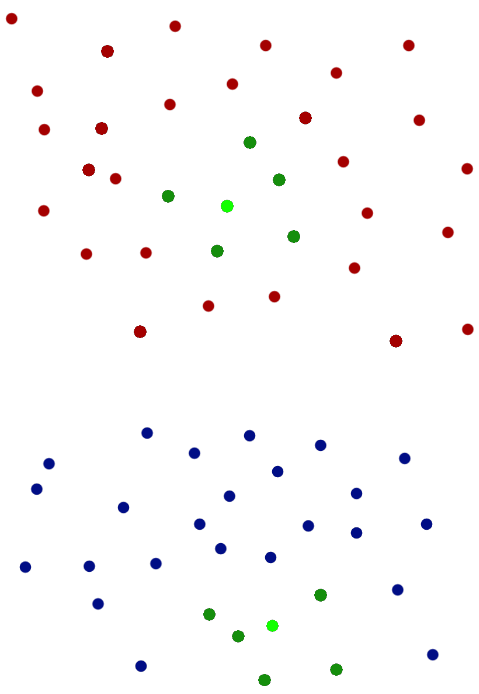

Figure 4.2: A successful embedding. The neighbours (dark green) of the query object (light green) are close together in both the observation space (top) and embedding space (bottom)

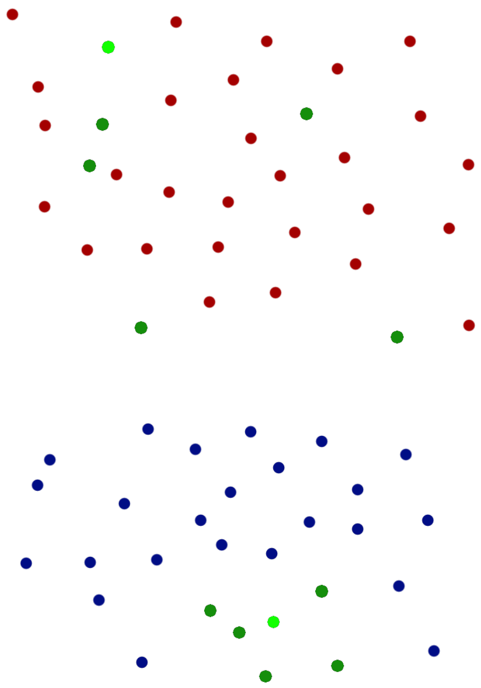

Figure 4.3: A poor embedding. The neighbours (dark green) of the query object (light green) in the embedding space (bottom) do not represent objects which are close together in the observation space (top). This indicates that information about which objects behave in similar ways has been lost during the embedding process.

of techniques which are best suited to various types of data. This allows us to produce a collection of low dimensional spaces which could potentially used for object recall. In all cases we made use of the default parameter values of the Matlab toolbox.

Results for all three datasets (primitive, complex and combined) are provided in Figure 4.4, Figure 4.5 and Figure 4.6 respectively. Results are rounded to integer values. As can be seen from these PCA and Sammon Mapping both result in the lowest error scores (as defined in section 4.6) with tSNE also having consistently low scores.

Vargha-Delaney A-test scores (Vargha & Delaney, 2000) were calculated and showed no significant difference between any of the views of the data we looked at. That is to say, no particular view of the data provided a significant benefit over any other once dimensionality reduction had been applied.

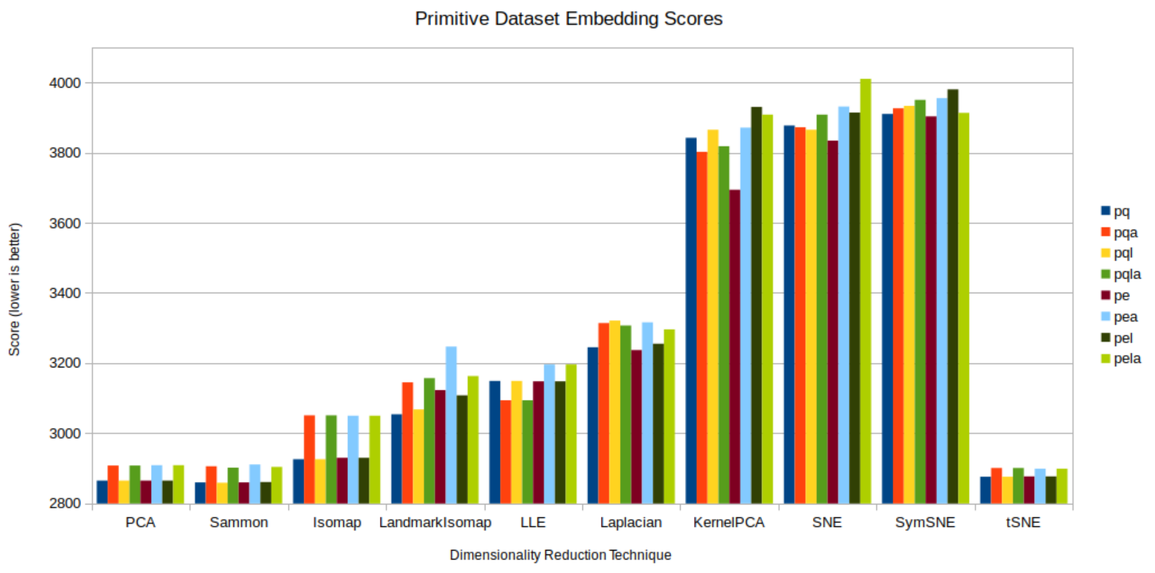

Figure 4.4: Embedding results for the Primitive dataset (lower scores are better). PCA, Sammon mapping, and tSNE consistently perform well. KernelPCA, SNE and SymSNE have high error scores in all cases when compared to the results of other methods.

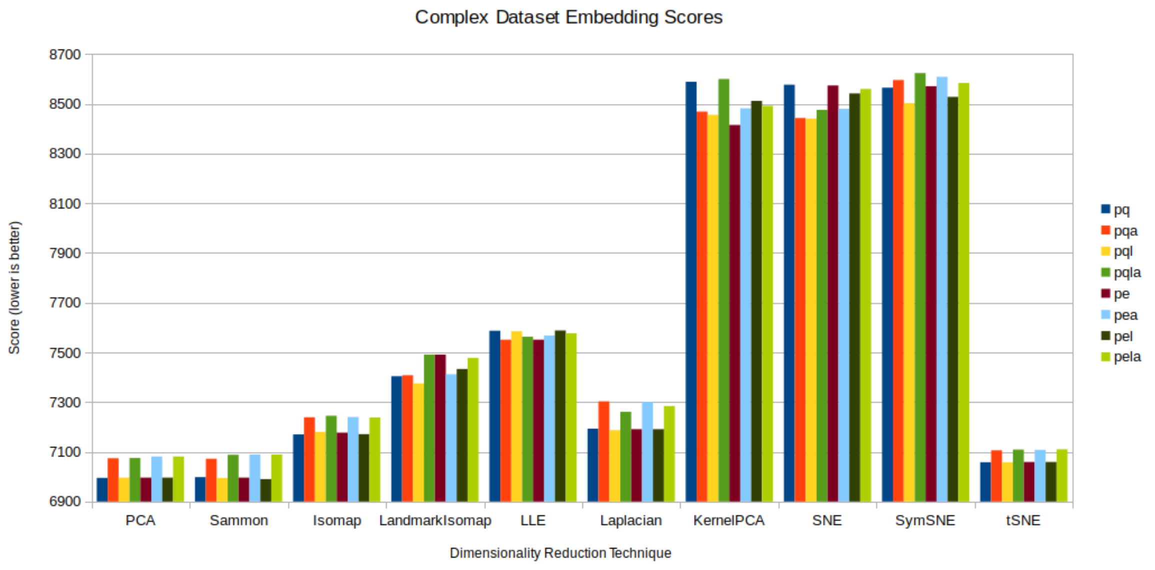

Figure 4.5: Embedding results for the Complex dataset (lower scores are better). As in Figure 4.4 PCA and Sammon mapping perform well. In this case, with more complex data, tSNE does not seem to perform as well as it does in the Primitive dataset. This suggests that, while tSNE performed well for simple shapes, it may be less applicable for real world applications. KernelPCA, SNE and SymSNE, again, all have high error scores.

4.8 Distribution of Object Recall Quality

Having identified PCA as the most appropriate technique for dimensionality reduction we now analyse the distributions of individual object recall for each view of the data when PCA is used.

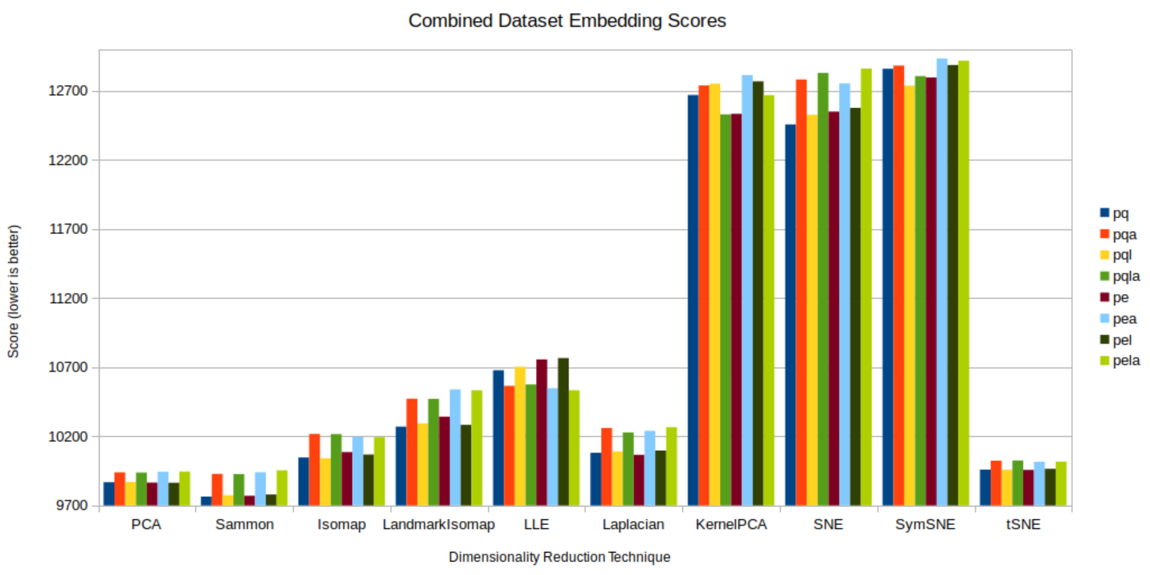

Figure 4.6: Embedding results for the Combined dataset (lower scores are better). As expected, based on the results presented in the individual data sets (Figure 4.4 and Figure 4.5), PCA and Sammon mapping perform best, followed by tSNE with KernelPCA, SNE and SymSNE performing worst.

	Primitive	Complex	Combined
PCA	0.092	0.168	0.295
Sammon	4.513	12.461	22.656
Isomap	0.462	1.372	2.989
Landmark Isomap	0.180	0.451	0.936
LLE	0.497	0.986	1.545
Laplacian	0.405	0.762	0.780
KernelPCA	0.140	0.298	0.484
SNE	16.986	36.327	58.334
SymSNE	16.073	35.353	56.476
tSNE	1.563	3.299	5.133

Table 4.2: Time taken (in seconds) for each technique to generate embeddings for each dataset. Times are calculated using the Matlab ‘tic’ and ‘toc’ commands when generating embeddings for all views of the respective datasets. Despite the quality of the embeddings generated (presented in Figure 4.4, Figure 4.5 and Figure 4.6) Sammon mapping is far more computationally expensive and so, in this case, we feel PCA is the most suitable option.

We also investigate the error distributions for each technique over the ‘pq’ view of the dataset.

Figure 4.7 shows the distribution of errors when each of the explored techniques is applied to the ‘pq’ view of the data. As can be seen more clearly here Sammon mapping and PCA have similar distributions of individual object recall error as well as there being very little difference between their overall error scores.

Figure 4.8 shows how PCA performs when applied to each of the eight views of the combined dataset. It can be seen more clearly here that the choice of orientation representation (Euler

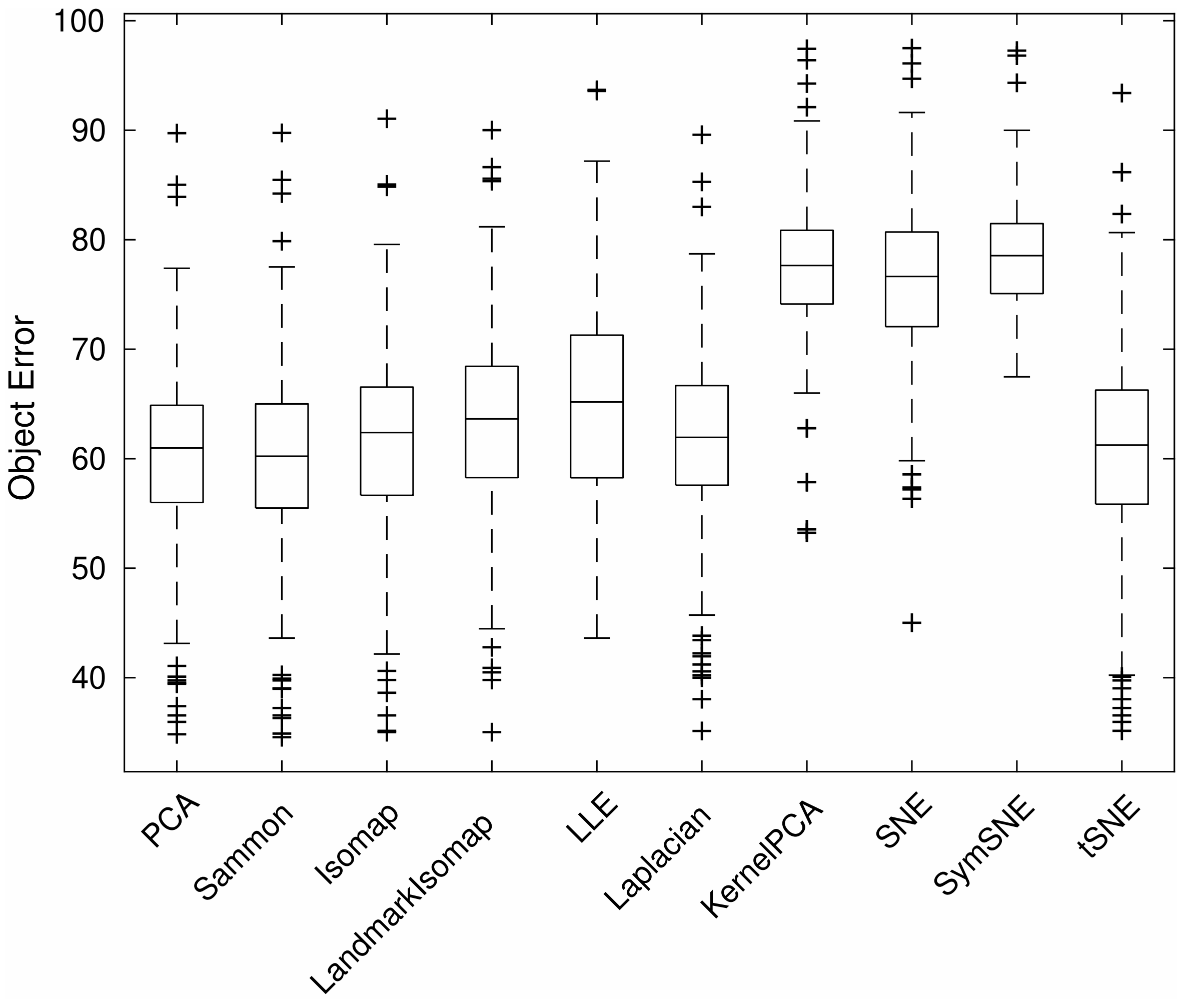

Figure 4.7: Distribution of individual object recall error using an embedding generated by various techniques on the 'pq' view of the combined dataset. A-test scores show no significant difference between PCA, Sammon mapping and tSNE or between KernelPCA, SNE and SymSNE.

angles vs quaternion) and the inclusion of linear and/or angular velocity makes very little difference to the quality of the generated embedding space. This is supported by A-test scores which also show little difference between the views.

4.9 Runtime of Dimensionality Reduction Techniques

As the target application for these techniques is recall onboard a mobile robot, which may have limited compute resources, we also analyse the run time of each technique. The values provided in Table 4.2 are the total times required, in seconds, to produce the embedding of all eight views of the data using each technique for each dataset.

It is not anticipated that embedding will need to be regenerated on a regular basis, however each time a new object is discovered and added to the memory it may be necessary to add the

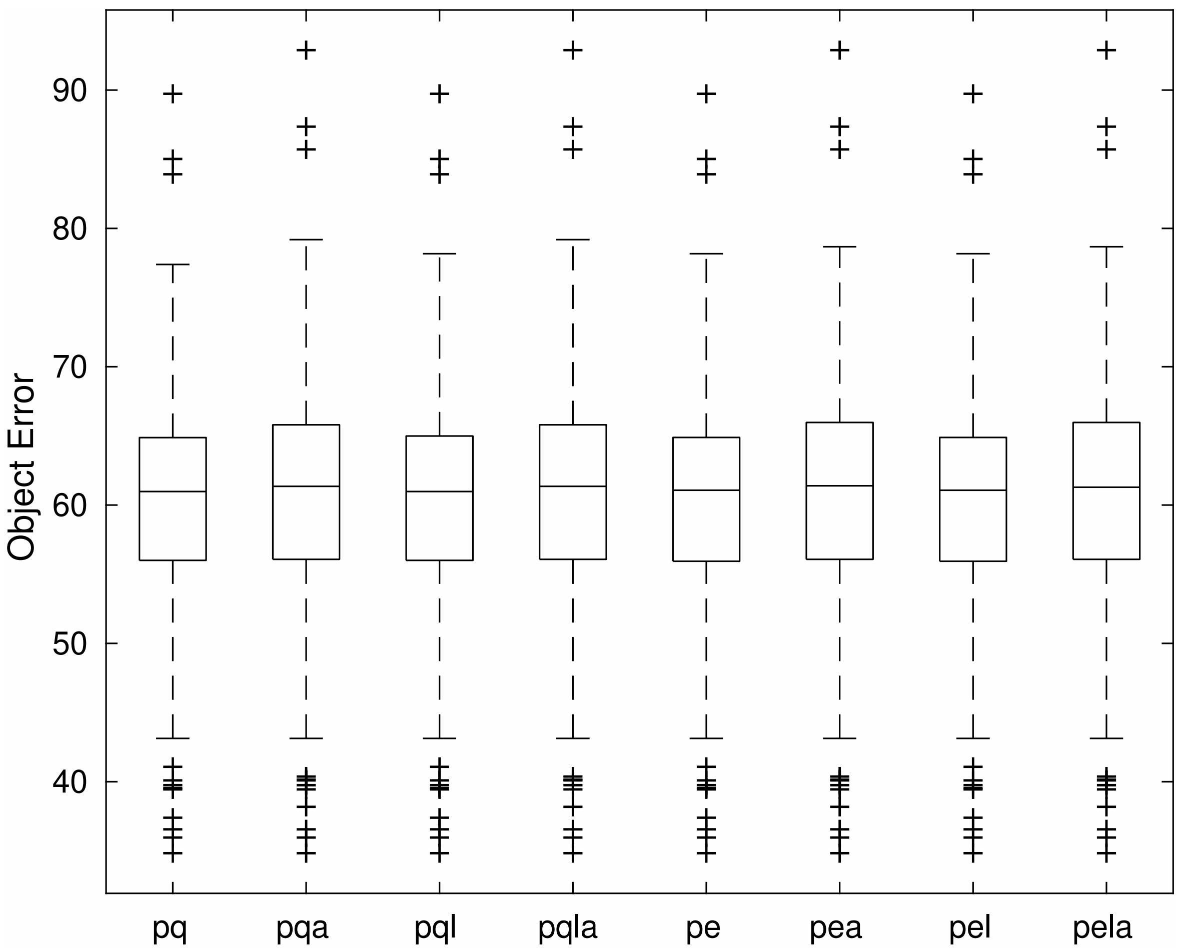

Figure 4.8: PCA error distribution when embeddings are generated from each view of the combined dataset. Here we see very little difference between the distributions of recall error. A-test scores show that there is only a small difference between all of the views.

object to the embedding, which requires complete recalculation for many techniques.

Although PCA and Sammon mapping produce the best results in individual object recall, with A-test scores showing no significant difference between them, it should be noted that the run time of Sammon mapping is approximately 2 orders of magnitude slower than that of PCA.

The increased run time of embedding generation suggests that the difference in error score between Sammon mapping and PCA is not significant enough for Sammon mapping to be considered a more appropriate technique in this case.

4.10 Summary

In this chapter we have presented a new dataset which can be used to benchmark the problem of object recall based on experience interactions. Prior to this work no benchmark set was available as the problem had not been investigated.

We have also presented the results of various dimensionality reduction techniques applied to this dataset for the purposes of object recall. We have shown that PCA performed on the position and orientation of a robot over multiple timesteps, including or excluding linear and angular velocity data, produces the highest quality embedding spaces of the techniques we have explored, and also requires the least computational time to produce. These results provide a basis for future work which could allow for scene recall techniques which could be used to accelerate expensive planning processes when robots encounter new problems.

We have also shown that the addition of information is not necessarily beneficial to producing a higher quality embedding space, and that more complex techniques do not necessarily result in higher quality embeddings.

We believe that this is an incredibly interesting area of research, and one which could provide tremendous benefit to autonomous robotic systems where planning is required. This area could provide even greater utility in scenarios where robots are deployed for extended periods and are expected to learn about and adapt to their environment in an unsupervised manner.

Chapter 5

Summary

In this thesis we have presented an analysis of multiple dimensionality reduction techniques which can be used to produce relatively low dimensional embedding spaces for efficient recall of individual objects. These embedding spaces are designed in such a way that knowledge transfer can take place between objects, as the spaces are optimised to place objects together based on the way they react to manipulation by a robot or other agent. Being able to recall objects based on similarity of how those objects can be manipulated allows for knowledge transfer of potentially useful actions between collections of recognised objects which are positioned in similar layouts and has not previously been explored.

We have also proposed several changes to best first search techniques, primarily based around Monte-Carlo Tree Search. These changes allow the application of best first search techniques to more complex search problems and potentially make some previously compuationally intractable problems solvable by manipulating the way the search space is explored, and by biasing the search process towards areas of the seach space which were believed to be fruitful when performing previous searches.

5.1 Contributions

The primary contributions from the work carried out in the production of this thesis are as follows:

5.1.1 Object Embedding Dataset

Prior to the work carried out and presented here the problem of recalling objects based on the way the can be manipulated and interacted with, to the best of our knowledge, had not been explored. As a result there were no benchmark datasets or results to be used when this work began. To this end, a dataset was developed which was used for the experiments presented in this work.

The dataset developed to carry out this work has been made publically available and can be used in any future work to produce similar recall techniques.

5.1.2 Analysis of Dimensionality Reduction Techniques for Object Recall

The object recall dataset developed and presented above has been used to explore the applicability of various dimensionality reduction techniques to the problem of individual object recall. We have analysed the performance of these techniques to several views of the objects in the embedding dataset and shown that Principal Components Analysis performed on the position and orientation of a robot and object resulted in the strongest performing embedding of the dataset of the techniques investigated.

We have also shown that there is no significant difference between which form of orientation data is used (quaternion or Euler angle) and that the addition of linear and/or angular velocity data provided no significant benefit to embedding quality.

5.1.3 Transplanted Tree Search (TTS)

We have proposed changes to Monte-Carlo Tree Search which allow its application to single agent planning applications where a complete action sequence is required rather than knowledge of the next move to take. These changes also remove the possibility of producing a search tree which contains multiple routes to the same world state, and removes the potential of infinitely deep action sequences which do not modify the state of the world but do use up resources.

5.1.4 Experience Based Transplanted Tree Search (EBTTS)

Finally, we have presented a process we have termed "Experience Based Transplanted Tree Search" (EBTTS). EBTTS extends TTS to also take in to account previously solved or encountered problem to bias the search process of TTS towards areas of the search space which are believed to be more likely to contain valid solutions.

5.2 Future Work

There are many potential future directions in which this work could be taken. There are still many different dimensionality reduction techniques which may prove useful for object recall. These include neural network based techniques, which may be more applicable to online adaptation, as well as many extra techniques which are prevalent in the field of machine learning.

Building on the results presented here for single object recall it should also be possible to explore various techniques which allow individual object embeddings to be combined to allow scene recall to take place. Some proposed approaches to this problem have been presented here, however they were not explored in detail and could provide an exciting opportunity for later work.

Implementation of TTS and EBTTS, both proposed here, could be an extremely powerful tool, allowing for expensive search processes to take place on resource constrained platforms such as autonomous mobile robots. We believe that these techniques could prove to be incredibly fruitful in allowing autonomous multi robot systems to explore more complex and challenging environments than previously possible.

Appendix A

Bibliography

Bibliography

Andre, T., Neuhold, D. & Bettstetter, C. (2014), 'Coordinated multi-robot exploration: Out of the box packages for ROS', in 'Globecom Workshops (GC Wkshps), 2014', pp. 1457–1462.

Auer, P., Cesa-Bianchi, N. & Fischer, P. (2002), 'Finite-time Analysis of the Multiarmed Bandit Problem', *Machine Learning*, **47**(2-3), pp. 235–256, ISSN 0885-6125, 1573-0565.

Bailey, T., Nieto, J., Guivant, J., Stevens, M. & Nebot, E. (2006), 'Consistency of the EKF-SLAM Algorithm', in '2006 IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 3562–3568.

Bay, H., Tuytelaars, T. & Gool, L. V. (2006), 'SURF: Speeded Up Robust Features', in Leonardis, A., Bischof, H. & Pinz, A., eds., 'Computer Vision – ECCV 2006', Number 3951 in Lecture Notes in Computer Science, pp. 404–417, Springer Berlin Heidelberg, ISBN 978-3-540-33832-1 978-3-540-33833-8.

Belkin, M. & Niyogi, P. (2003), 'Laplacian eigenmaps for dimensionality reduction and data representation', *Neural Comput.*, **15**(6), pp. 1373–1396, ISSN 0899-7667.

Bellman, R. (1957), 'A markovian decision process', *Journal of Mathematics and Mechanics*, **6**(5), pp. 679–684, ISSN 00959057, 19435274.

Brechtel, S., Gindele, T. & Dillmann, R. d. (2013), 'Solving Continuous POMDPs: Value Iteration with Incremental Learning of an Efficient Space Representation', pp. 370–378.

Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S. & Colton, S. (2012), 'A Survey of Monte Carlo Tree Search Methods', *IEEE Transactions on Computational Intelligence and AI in Games*, **4**(1), pp. 1–43, ISSN 1943-068X.

Brunel, N. & van Rossum, M. C. W. (2007), 'Lapicque's 1907 paper: from frogs to integrate-and-fire', *Biological Cybernetics*, **97**(5), pp. 337–339, ISSN 1432-0770.

Carlone, L., Kaouk Ng, M., Du, J., Bona, B. & Indri, M. (2010), 'Rao-blackwellized particle filters multi robot slam with unknown initial correspondences and limited communication', in '2010 IEEE International Conference on Robotics and Automation', pp. 243–249.

Chaslot, G., Bakkes, E., Szita, I. & Spronck, P. (2008), 'Monte-Carlo Tree Search: A New Framework for Game AI', in 'In Proceedings of AIIDEC-08', pp. 216–217.

Conditt, J. (2015), 'Turn your home into a haunted house with AR game 'Night Terrors' (online: http://www.engadget.com/2015/04/30/night-terrors-horror-ar-game/)', .

Cook, J., Sutskever, I., Mnih, A. & Hinton, G. (2007), 'Visualizing similarity data with a mixture of maps', *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, pp. 67–74.

Couëtoux, A., Hoock, J.-B., Sokolovska, N., Teytaud, O. & Bonnard, N. (2011), 'Continuous Upper Confidence Trees', in Coello, C. A. C., ed., 'Learning and Intelligent Optimization', Number 6683 in Lecture Notes in Computer Science, pp. 433–445, Springer Berlin Heidelberg, ISBN 978-3-642-25565-6 978-3-642-25566-3.

Cunningham, A., Paluri, M. & Dellaert, F. (2010), 'Ddf-sam: Fully distributed slam using constrained factor graphs', in '2010 IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 3025–3030.

Davison, A., Reid, I., Molton, N. & Stasse, O. (2007), 'MonoSLAM: Real-Time Single Camera SLAM', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(6), pp. 1052–1067, ISSN 0162-8828.

Drost, B., Ulrich, M., Navab, N. & Ilic, S. (2010), 'Model globally, match locally: Efficient and robust 3d object recognition', in '2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 998–1005.

Eliazar, A. & Parr, R. (2004), 'DP-SLAM 2.0', in '2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04', volume 2, pp. 1314–1320 Vol.2.

Engelson, S. & McDermott, D. (1992), 'Error correction in mobile robot map learning', in ', 1992 IEEE International Conference on Robotics and Automation, 1992. Proceedings', pp. 2555–2560 vol.3.

Ferworn, A., Herman, S., Tran, J., Ufkes, A. & Mcdonald, R. (2013), 'Disaster Scene Reconstruction: Modeling and Simulating Urban Building Collapse Rubble Within a Game Engine', in 'Proceedings of the 2013 Summer Computer Simulation Conference', SCSC '13, pp. 18:1–18:6, Society for Modeling & Simulation International, Vista, CA, ISBN 978-1-62748-276-9.

Forstmann, S. (2016), ‘Fast quadric mesh simplification’, *GitHub - https://github.com/sp4cerat/Fast-Quadric-Mesh-Simplification*.

Gerkey, B. P., Vaughan, R. T. & Howard, A. (2003), ‘The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems’, in ‘In Proceedings of the 11th International Conference on Advanced Robotics’, pp. 317–323.

Gilchrist, I. D., Brown, V. & Findlay, J. M. (1997), ‘Saccades without eye movements’, *Nature*, **390**(6656), p. 130.

Gildert, N., Millard, A. G., Pomfret, A. & Timmis, J. (2018), ‘The need for combining implicit and explicit communication in cooperative robotic systems’, *Frontiers in Robotics and AI*, **5**, p. 65.

Guivant, J. & Nebot, E. (2001), ‘Optimization of the simultaneous localization and map-building algorithm for real-time implementation’, *IEEE Transactions on Robotics and Automation*, **17**(3), pp. 242–257, ISSN 1042-296X.

Gálvez-López, D., Salas, M., Tardós, J. D. & Montiel, J. M. M. (2015), ‘Real-time Monocular Object SLAM’, .

Hinton, G. E. & Roweis, S. T. (2003), ‘Stochastic neighbor embedding’, in Becker, S., Thrun, S. & Obermayer, K., eds., ‘Advances in Neural Information Processing Systems 15’, pp. 857–864, MIT Press.

Hopfield, J. J. (1982), ‘Neural networks and physical systems with emergent collective computational abilities’, *Proceedings of the National Academy of Sciences*, **79**(8), pp. 2554–2558, ISSN 0027-8424, 1091-6490.

Hotelling, H. (1933), ‘Analysis of a complex of statistical variables into principal components’, *Journal of Educational Psychology*, **24**(6), pp. 417–441, ISSN 1939-2176(Electronic);0022-0663(Print).

Howard, A. (2006), ‘Multi-robot Simultaneous Localization and Mapping using Particle Filters’, *The International Journal of Robotics Research*, **25**(12), pp. 1243–1256, ISSN 0278-3649, 1741-3176.

Hsu, D., Lee, W. S. & Rong, N. (2007), ‘What makes some POMDP problems easy to approximate?’, .

Kasper, A., Xue, Z. & Dillmann, R. (2012), ‘The kit object models database: An object model database for object recognition, localization and manipulation in service robotics’, *The International Journal of Robotics Research*, **31**(8), pp. 927–934.

Kocsis, L. & Szepesvári, C. (2006), ‘Bandit Based Monte-Carlo Planning’, in Fürnkranz, J., Scheffer, T. & Spiliopoulou, M., eds., ‘Machine Learning: ECML 2006’, Number 4212 in Lecture Notes in Computer Science, pp. 282–293, Springer Berlin Heidelberg, ISBN 978-3-540-45375-8 978-3-540-46056-5.

Kroger, T., Finkemeyer, B., Winkelbach, S., Eble, L.-O., Molkenstruck, S. & Wahl, F. (2008), ‘A manipulator plays Jenga’, *IEEE Robotics Automation Magazine*, **15**(3), pp. 79–84, ISSN 1070-9932.

Le, Q. V. & Mikolov, T. (2014), ‘Distributed representations of sentences and documents’, *arXiv preprint arXiv:1405.4053*.

Leonard, J. & Durrant-Whyte, H. (1991), ‘Simultaneous map building and localization for an autonomous mobile robot’, in ‘IEEE/RSJ International Workshop on Intelligent Robots and Systems ’91. ’Intelligence for Mechanical Systems, Proceedings IROS ’91’, pp. 1442–1447 vol.3.

Leonard, J. J. & Feder, H. J. S. (2000), ‘A computationally efficient method for large-scale concurrent mapping and localization’, in Hollerbach, J. M. & Koditschek, D. E., eds., ‘Robotics Research’, pp. 169–176, Springer London, London, ISBN 978-1-4471-0765-1.

Lowe, D. (1999), ‘Object recognition from local scale-invariant features’, in ‘The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999’, volume 2, pp. 1150–1157 vol.2.

Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P. & Milford, M. J. (2016), ‘Visual place recognition: A survey’, *IEEE Transactions on Robotics*, **32**(1), pp. 1–19.

Michel, O. (2004), ‘Webots: Professional Mobile Robot Simulation’, *International Journal of Advanced Robotic Systems*, **1**(1), pp. 39–42.

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013), ‘Efficient Estimation of Word Representations in Vector Space’, *arXiv:1301.3781 [cs]*, arXiv: 1301.3781.

Millard, A. G., Timmis, J. & Winfield, A. F. T. (2014), ‘Towards exogenous fault detection in swarm robotic systems’, in Natraj, A., Cameron, S., Melhuish, C. & Witkowski, M., eds., ‘Towards Autonomous Robotic Systems’, pp. 429–430, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN 978-3-662-43645-5.

Montemerlo, M., Thrun, S., Koller, D. & Wegbreit, B. (2003), ‘FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges’, in ‘In Proc. of the Int. Conf. on Artificial Intelligence (IJCAI’, pp. 1151–1156.

Ornan, O. & Degani, A. (2013), 'Toward autonomous disassembling of randomly piled objects with minimal perturbation', in '2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 4983–4989.

Paz, L. M., Guivant, J., Tardós, J. D. & Neira, J. (2008), 'Divide and Conquer: EKF SLAM in O(n)', *IEEE TRANSACTIONS ON ROBOTICS*.

Pei, F., Wu, M. & Zhang, S. (2014), 'Distributed SLAM Using Improved Particle Filter for Mobile Robot Localization', *The Scientific World Journal*, **2014**, p. e239531, ISSN 2356-6140.

Pillai, S. & Leonard, J. (2015), 'Monocular SLAM Supported Object Recognition', *arXiv:1506.01732 [cs]*, arXiv: 1506.01732.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Caro, G. D., Ducatelle, F., Birattari, M., Gambardella, L. M. & Dorigo, M. (2012), 'ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems', *Swarm Intelligence*, **6**(4), pp. 271–295, ISSN 1935-3812, 1935-3820.

Plate, T. A. (2003), Holographic Reduced Representation: Distributed Representation for Cognitive Structures, University of Chicago Press, Stanford, Calif, new edition edition edition, ISBN 978-1-57586-430-3.

Redpath, R., Timmis, J. & Trefzer, M. (2017), 'Object recall using an experience database to accelerate robot action planning', in '2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 1566–1571, ISSN 2153-0866.

Rohmer, E., Singh, S. & Freese, M. (2013), 'V-REP: A versatile and scalable robot simulation framework', in '2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 1321–1326.

Roweis, S. T. & Saul, L. K. (2000), 'Nonlinear dimensionality reduction by locally linear embedding', *SCIENCE*, **290**, pp. 2323–2326.

Russell, S. & Norvig, P. (2013), Artificial Intelligence: A Modern Approach, Pearson, Harlow, third edition, ISBN 978-1-292-02420-2.

Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H. & Davison, A. J. (2013), 'SLAM++: Simultaneous Localisation and Mapping at the Level of Objects', pp. 1352–1359, IEEE, ISBN 978-0-7695-4989-7.

Sammon, J. W. (1969), 'A nonlinear mapping for data structure analysis', *IEEE Transactions on Computers*, **C-18**(5), pp. 401–409, ISSN 0018-9340.

Schölkopf, B., Smola, A. & Müller, K.-R. (1998), 'Nonlinear component analysis as a kernel eigenvalue problem', *Neural Comput.*, **10**(5), pp. 1299–1319, ISSN 0899-7667.

Shaojun, C. (2014), Online POMDP Planning for Vehicle Navigation in Densely Populated Area, Thesis.

Sharkey, A. J. C. (2007), 'Swarm robotics and minimalism', *Connection Science*, **19**(3), pp. 245–260.

Silva, V. D. & Tenenbaum, J. B. (2003), 'Global versus local methods in nonlinear dimensionality reduction', in 'Advances in Neural Information Processing Systems 15', pp. 705–712, MIT Press.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T. & Hassabis, D. (2016), 'Mastering the game of go with deep neural networks and tree search', *Nature*, **529**, pp. 484–503.

Silver, D. & Veness, J. (2010), 'Monte-Carlo Planning in Large POMDPs', pp. 2164–2172.

Smith, R., Self, M. & Cheeseman, P. (1990), 'Estimating Uncertain Spatial Relationships in Robotics', in Cox, I. J. & Wilfong, G. T., eds., 'Autonomous Robot Vehicles', pp. 167–193, Springer New York, ISBN 978-1-4613-8999-6 978-1-4613-8997-2.

Smith, R. C. & Cheeseman, P. (1986), 'On the Representation and Estimation of Spatial Uncertainty', *The International Journal of Robotics Research*, **5**(4), pp. 56–68, ISSN 0278-3649, 1741-3176.

Tenenbaum, J. B., Silva, V. d. & Langford, J. C. (2000), 'A global geometric framework for nonlinear dimensionality reduction', *Science*, **290**(5500), pp. 2319–2323, ISSN 0036-8075.

Torgerson, W. S. (1952), 'Multidimensional scaling: I. theory and method', *Psychometrika*, **17**(4), pp. 401–419, ISSN 1860-0980.

van der Maaten, L. & Hinton, G. E. (2008), 'Visualizing high-dimensional data using t-sne', *Journal of Machine Learning Research*, **9**, pp. 2579–2605.

van der Maaten, L. J. P., Postma, E. O. & van den Herik, H. J. (2009), 'Dimensionality reduction: A comparative review', Technical report, Tilburg University.

Vargha, A. & Delaney, H. (2000), 'A critique and improvement of the cl common language effect size statistics of mcgraw and wong', *Journal of Educational and Behavioral Statistics*, **25**(2), pp. 101–132, ISSN 1076-9986.

Vaughan, R. & Zuluaga, M. (2006), 'Use Your Illusion: Sensorimotor Self-simulation Allows Complex Agents to Plan with Incomplete Self-knowledge', in Nolfi, S., Baldassarre, G., Calabretta, R., Hallam, J. C. T., Marocco, D., Meyer, J.-A., Miglino, O. & Parisi, D., eds., ‘From Animals to Animats 9’, Number 4095 in Lecture Notes in Computer Science, pp. 298–309, Springer Berlin Heidelberg, ISBN 978-3-540-38608-7 978-3-540-38615-5.

Wang, Y., Won, K. S., Hsu, D. & Lee, W. S. (2012), ‘Monte Carlo Bayesian Reinforcement Learning’, *arXiv:1206.6449 [cs, stat]*, arXiv: 1206.6449.

Weitnauer, E., Haschke, R. & Ritter, H. (2010), ‘Evaluating a Physics Engine as an Ingredient for Physical Reasoning’, in Ando, N., Balakirsky, S., Hemker, T., Reggiani, M. & Stryk, O. v., eds., ‘Simulation, Modeling, and Programming for Autonomous Robots’, Number 6472 in Lecture Notes in Computer Science, pp. 144–155, Springer Berlin Heidelberg, ISBN 978-3-642-17318-9 978-3-642-17319-6.

Winfield, A. F. T., Blum, C. & Liu, W. (2014), ‘Towards an Ethical Robot: Internal Models, Consequences and Ethical Action Selection’, in Mistry, M., Leonardis, A., Witkowski, M. & Melhuish, C., eds., ‘Advances in Autonomous Robotics Systems’, Number 8717 in Lecture Notes in Computer Science, pp. 85–96, Springer International Publishing, ISBN 978-3-319-10400-3 978-3-319-10401-0.