

Human inspired robotic path planning and heterogeneous robotic mapping

by

Henry Williams

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the requirements for the degree of
Doctor of Philosophy

Victoria University of Wellington
2016

Abstract

One of the biggest challenges facing robotics is the ability for a robot to autonomously navigate real-world unknown environments and is considered by many to be a key prerequisite of truly autonomous robots. Autonomous navigation is a complex problem that requires a robot to solve the three problems of navigation: localisation, goal recognition, and path-planning. Conventional approaches to these problems rely on computational techniques that are inherently rigid and brittle. That is, the underlying models cannot adapt to novel input, nor can they account for all potential external conditions, which could result in erroneous or misleading decision making.

In contrast, humans are capable of learning from their prior experiences and adapting to novel situations. Humans are also capable of sharing their experiences and knowledge with other humans to bootstrap their learning. This is widely thought to be an important part of the success of humanity by allowing high-fidelity transmission of information and skills between individuals, facilitating cumulative knowledge gain. Furthermore, human cognition is influenced by internal emotion states. Historically considered to be a detriment to a person's cognitive process, recent research is regarding emotions as a beneficial mechanism in the decision making process by facilitating the transfer of simple, but high-impact information.

Human created control approaches are inherently rigid and cannot account for the complexity of behaviours required for autonomous navigation. The proposed thesis is that cognitive inspired mechanisms can address limitations in current robotic navigation techniques by allowing

robots to autonomously learn beneficial behaviours from interacting in their environment. The first objective is to enable the sharing of navigation information between heterogeneous robotic platforms. The second objective is to add flexibility to rigid path-planning approaches by utilising emotions as low-level but high-impact behavioural responses.

Inspired by cognitive sciences, a novel cognitive mapping approach is presented that functions in conjunction with current localisation techniques. The cognitive mapping stage utilises an Anticipatory Classifier System (ACS) to learn the novel Cognitive Action Map (CAM) of *decision points*, areas in which a robot must determine its next *action* (direction of travel). These physical actions provide a shared means of understanding the environment to allow for the transferring of learned navigation information.

The presented cognitive mapping approach has been trained and evaluated on real-world robotic platforms. The results show the successful sharing of navigation information between two heterogeneous robotic platforms with different sensing capabilities. The results have also demonstrated the novel contribution of autonomously sharing navigation information between a range-based (GMapping) and vision-based (RatSLAM) localisation approach for the first time. The advantage of sharing information between localisation techniques allows an individual robotic platform to utilise the best fit localisation approach for its sensors while still being able to provide useful navigation information for robots with different sensor types.

Inspired by theories on natural emotions, this work presents a novel emotion model designed to improve a robot's navigation performance through learning to adapt a rigid path-planning approach. The model is based on the concept of a bow-tie structure, linking emotional reinforcers and behavioural modifiers through intermediary emotion states. An important function of the emotions in the model is to provide a compact set of high-impact behaviour adaptations, reducing an otherwise tangled web of

stimulus-response patterns. Crucially, the system learns these emotional responses with no human pre-specifying the behaviour of the robot, hence avoiding human bias.

The results of training the emotion model demonstrate that it is capable of learning up to three emotion states for robotic navigation without human bias: *fear*, *apprehension*, and *happiness*. The fear and apprehension responses slow the robot's speed and drive the robot away from obstacles when the robot experiences *pain*, or is uncertain of its current position. The happiness response increases the speed of the robot and reduces the safety margins around obstacles when pain is absent, allowing the robot to drive closer to obstacles. These learned emotion responses have improved the navigation performance of the robot by reducing collisions and navigation times, in both simulated and real-world experiments. The two emotion model (fear and happiness) improved performance the most, indicating that a robot may only require two emotion states (fear and happiness) for navigation in common, static domains.

Acknowledgements

I would first like to thank my supervisors Will N. Browne and Dale Carnegie for the invaluable advice over the course of both my PhD and undergraduate studies. I greatly appreciate the amount of time and effort you have put into helping me with my research, and the feedback on the papers written throughout. You both have the patience of saints for teaching me the ways of using grammar properly. In saying that, thank you Louis McArdell for spending some of your free time on proof reading this thesis.

I am grateful to Victoria University of Wellington (VUW) for awarding me the Victoria PhD Scholarship and to Will N. Browne and Mengjie Zhang for supporting my PhD scholarship application. Without the financial support I would not have been able to conduct this research.

I would also like to thank the technical staff at VUW for the support in setting up and handling the robotic platforms. Thank you in particular to Brett Ryan for spending time away from your masters to help develop and construct Ownbot, special mention to Robby Lopez and Jeremy Ley for assisting.

Last but not least, I give thanks to my friends and family for supporting me during my PhD. Especially to those who have tolerated my impassioned and long winded discussions of robotics and machine learning. Your excellent listening skills have helped me improve my understanding of the work I have conducted and helped me explain my ideas with greater clarity.

Contents

Acknowledgements	v
Glossary	xv
1 Introduction	1
1.1 Motivations and Objectives	5
1.1.1 Cognitive action mapping	5
1.1.2 Adaptive path-planning through emotions	7
1.2 Thesis outline	10
2 Background	13
2.1 SLAM	14
2.1.1 Statistical based SLAM	14
2.1.2 Appearance based SLAM	17
2.1.3 Active SLAM	18
2.2 Robotic navigation	18
2.3 Bow-tie models	22
2.4 Emotion models	24
2.5 Machine learning	29
2.5.1 Reinforcement learning	30
2.5.2 Challenges of RL in robotics	32
2.5.3 Learning Classifier Systems	33
2.6 XCS: Accuracy based LCS	38

2.6.1	XCS classifier	38
2.6.2	Learning parameters	39
2.6.3	Parameter update	41
2.7	ACS: Anticipatory classifier system	42
2.7.1	ACS classifier	43
2.7.2	Learning parameters	43
2.7.3	Parameter update	44
2.8	Robot operating system	45
2.9	Summary	47
3	Cognitive Action Mapping SLAM	49
3.1	CAM-SLAM model	51
3.1.1	Decision points	51
3.1.2	Mental mapping: SLAM	54
3.2	Anticipatory Classifier System	54
3.3	Implementation	57
3.3.1	Action selection	61
3.3.2	Parameter update	61
3.4	Experimental setup	63
3.4.1	Robotic platforms	63
3.4.2	Environment	65
3.5	Experiments	67
3.5.1	Experiment one: Learning a CAM	67
3.5.2	Experiment two: Benchmark algorithm	70
3.5.3	Experiment three: Learning a change in the environ- ment	70
3.5.4	Experiment four: Sharing navigation information through CAM-SLAM	70
4	Cognitive Action Mapping SLAM Results	73
4.1	Learning a CAM with Ownbot	74
4.1.1	Experiment one: Ownbot	75

4.2	Learning a CAM with Pioneer	80
4.2.1	Experiment one: Pioneer	80
4.2.2	Pioneer: GMapping mental mapping stage	80
4.2.3	Pioneer: RatSLAM mental mapping stage	81
4.2.4	Summary of initial learning	83
4.3	Benchmark Algorithm experiment	86
4.4	Learning a change in the environment	86
4.5	Sharing navigation information	90
4.5.1	Same sensor type, different morphology experiment	91
4.5.2	Different sensor, different morphology experiments	95
4.5.3	Summary of shared learning	99
4.6	Summary	99
5	Emotion Inspired Path Planning	101
5.1	Emotion model	103
5.1.1	Reinforcers	105
5.1.2	Modifiers	108
5.2	Anthropomorphic emotions	111
5.3	Implementation	113
5.3.1	Reinforcer Input	116
5.3.2	Reinforcer-Emotion	117
5.3.3	Emotion-Modifier	118
5.3.4	Update	119
5.4	Experimental setup	120
5.4.1	Experiment One: Simulated emotion model training	122
5.4.2	Experiment two: Learn-ability of the system	124
5.4.3	Experiment three: Emotion model path-planning performance	125
5.4.4	Experiment four: Real-world emotion model training	125
6	Emotion Inspired Path Planning Results	129
6.1	Simulated emotion model training	130

6.1.1	Two Emotions	131
6.1.2	Three Emotions	136
6.1.3	Four Emotions	139
6.2	Learn-ability of the system	140
6.3	Emotion model path-planning performance	142
6.4	Real-world emotion model training	145
6.5	Discussion	148
6.6	Summary	151
7	Conclusions	153
7.1	Cognitive action mapping	153
7.2	Emotion Inspired Path Planning	155
7.3	Summary of contributions	157
7.4	Future Work	158
7.5	Publications	160
7.6	Final Summary	161
	Bibliography	163

List of Figures

2.1	Flow diagram of SLAM	15
2.2	Metric versus topological mapping representations	20
2.3	Many inputs with high variability are filtered through the 'knot' of the bow-tie. The knot then determines the output or response of the system.	23
2.4	Roll's frame work of emotions	27
2.5	Flow diagram of a LCS	35
3.1	Flow diagram of CAM-SLAM	52
3.2	An example of a decision point	53
3.3	Representation of a classifier in a map	55
3.4	Pioneer and Ownbot	66
3.5	Ground truth of level two of the Cotton building	68
3.6	Ground truth of level three of the Cotton building	69
4.1	CAM learned by Ownbot on the third floor of Cotton	78
4.2	Mental map by Ownbot on the third floor of Cotton	79
4.3	CAM learned by Pioneer on the second floor of Cotton	81
4.4	Mental map by Pioneer on second floor of Cotton	82
4.5	Image based CAM learned by Pioneer on second floor of Cotton	84
4.6	Example of an image based decision point	85
4.7	Human generated map of the third floor of Cotton	87

4.8	Subsection of a CAM	89
4.9	LIDAR based CAM learned through sharing	92
4.10	LIDAR and Image based CAM learned through sharing	96
4.11	LIDAR and Image based CAM learned through sharing	98
5.1	Reinforcer-Emotion-Modifier bow-tie structure	104
5.2	Co-operative model of emotions and modifiers	105
5.3	Navigation Stack mapping example	109
5.4	Example of classifiers in LCS	115
5.5	Flow diagram of emotion system	116
5.6	Simulated Willow Garage office	123
5.7	Simulated benchmark course	126
5.8	Real-world benchmark course	127
6.1	Behavioural response of learned emotions	132
6.2	Bow-tie example of emotion one	133
6.3	Bow-tie example of emotion two (A)	135
6.4	Bow-tie example of emotion two (B)	135
6.5	Classifier fitness in each emotion model	140
6.6	Modifier fitness in each emotion model	141
6.7	Simulated comparison of time between each checkpoint	143
6.8	Simulated comparison of collisions between each checkpoint	144
6.9	Real-world (R) comparison of time between each checkpoint	147
6.10	Real-world (R) comparison of collisions between each checkpoint	148
6.11	Real-world (R) comparison of classifier and modifier fitness	149

List of Tables

3.1	Example of the environmental state as a classifier condition. Classifier one is an example of a classifier learned by a range based robot, and classifier two is an example from a vision based robot	55
4.1	Classifier population learned to lead to the goal location on the third floor of Cotton.	76
4.2	Average number of matches for Ownbot Classifiers across 10 trials	93
4.3	Average number of new classifiers created by the Pioneer for each of Ownbot's classifiers across 10 trials	94
5.1	Modifier Example, showing default values from the Navigation stack.	111
5.2	Emotions and reinforcers that could potentially trigger them.	112
6.1	Example of two emotions with different labels that would be clustered into a single emotion state.	131
6.2	Learned most fit emotion to modifier mapping averaged over 30 trials: two emotions	132
6.3	Example of the consistent Reinforcer-Emotion classifiers for the emotion model trained with two emotion states	134
6.4	Learned most fit emotion to modifier mapping averaged over 30 trials: three emotions	137

6.5	Example of the consistent Reinforcer-Emotion classifiers for the emotion model trained with three emotion states	138
6.6	Learned most fit emotion to modifier mapping averaged over 30 trials: four emotions	139
6.7	Learned most fit emotion to modifier mapping: real-world experiment	145
6.8	Example of the consistent Reinforcer-Emotion classifiers for the emotion model trained with two emotion states: real-world	146

Glossary

ACS Anticipatory Classifier System.

CAM Cognitive Action Map.

CAM-SLAM Cognitive Action Mapping SLAM.

GA Genetic Algorithm.

LCS Learning Classifier System.

RL Reinforcement Learning.

ROS Robot Operating System.

SLAM Simultaneous Localisation and Mapping.

VUW Victoria University of Wellington.

XCS Accuracy Based Classifier System.

XCSCFC XCS using Code Fragmented Conditions.

Chapter 1

Introduction

The past couple of decades have seen extensive progress towards making autonomous robots a reality. A very recent show case of just how far autonomous robotics has come is the current DARPA robotics challenge¹. Teams demonstrated humanoid robots completing a series of autonomous challenges, including climbing stairs, navigating obstacle rich zones, and physically driving cars. Although the winning robots were impressive they were still slow, prone to errors, and expensive. Autonomous mobile robotic research has come a long way, but still has a long way to go before robots are as capable of reliably navigating and operating in the real-world as effectively as humans do.

One of the biggest challenges facing mobile robotics is the ability for a robot to autonomously navigate unknown real-world environments and is considered by many to be one of the key prerequisite of truly autonomous robots [26], citefinn2012. Of course there are still many challenges to overcome outside of the task of navigation requiring different forms of non-mobile robotics platforms, e.g optimal kinematics or an assembly line robot. The focus of this research is on mobile robots and will be discussed in the general term 'robot' in this thesis.

The ability for a robot to be placed at an unknown location in an un-

¹<http://www.theroboticschallenge.org/>

known environment and then have it build a map, using only its own sensory observations, to simultaneously navigate with, would help make such a robot autonomous [26]. A solution to the navigation problem would be of great value in a range of applications where absolute position or precise map information is unavailable. This includes, amongst others, autonomous planetary exploration, sub-sea autonomous vehicles, autonomous air-borne vehicles, and autonomous all-terrain vehicles in tasks such as mining and construction. In order to achieve this level of autonomy a robot must solve the three complex tasks of navigation [89].

Where am I? The robot must know where it is within the environment in order to make a useful decision. Finding the whereabouts of a robot is termed *localisation*.

Where am I going? In order to fulfil a given task a robot must know where it is going, this is known as *goal recognition*.

How do I get there? Once the robot knows where it is, and where it is going, it has to determine how it is going to get to the goal. This is known as *path-planning*.

Obtaining the localisation of the robot ('Where am I?') is a non-trivial problem, as there are many possible states available in real-world problems. Similar to humans, a robot requires an accurate map of its environment to localise from. However, to generate an accurate map within an unknown environment a robot must be localised. Simultaneous Localisation and Mapping (SLAM), see section 2.1, is the technique that seeks to allow a robot to autonomously localise itself within unknown environments. The main advantage of SLAM is that it eliminates the need for human generated infrastructures or *a priori* knowledge of the environment. This allows applications where map gathering is impractical, avoiding slow human map building and negating any human bias.

The problem of goal recognition is specified on a task by task basis, generally by a human operator. The robot is then required to determine

how it will achieve the specified goal. In robotic navigation the robot is typically given a goal location or goal target it must find within an environment. For example, searching for people in a disaster zone.

Path planning ('How do I get there?') is an important issue as it allows a robot to get from point A to point B within its environment, see section 2.2. For a robot to successfully path plan to a goal location it must have a map of its environment and know where the goal location is within the map. This becomes complex within real world environments where goal locations are not easily defined or known and maps are unavailable, making path-planning dependant on solving the localisation problem. For example, a rescue robot within a disaster zone looking for survivors, the 'goal location' of survivors is unknown and an accurate map of the environment is not going to be available.

Current approaches for solving these problems have advanced in the past couple of decades and have been shown to be effective in a variety of environments, see section 2. However, these traditional techniques rely on rigid computational approaches. Rigid computational models lack the abilities to adapt to novel environments or to improve their behaviour over time [77]. Although certain techniques have been designed to adapt to specific conditions within an environment, the term rigid is regarding a path-planning approaches lack of learning or adaptability to novel input the system is not designed for *a priori*.

This rigidity makes computational approaches inherently brittle in that underlying models cannot account for all potential external conditions or relevant variables which could result in erroneous or misleading decision making [91]. Rigidity can be lessened by manually optimising performance in a diverse set of environments, but this may result in a 'Jack of all trades, master of none' system that compromises performance in one type of environment to achieve satisfactory performance in another.

In contrast, humans are capable of not only reasoning about their environment but also learning from prior experiences and updating under-

lying decision making processes. This allows a person to perform in a variety of environments or conditions by adapting to novel experiences. Furthermore, people are capable of sharing their experiences and knowledge with the community, enabling a person to learn from the experiences of others. Teaching, alongside imitation, is widely thought to underlie the success of humanity by allowing high-fidelity transmission of information, skills, and technology between individuals, facilitating cumulative knowledge gain [37]. The communal benefit provides a greater understanding of a problem or environment, and reduces the risk to an individual exploring a new terrain on their own [36].

The field of cognitive robotics seeks to give robots high level cognitive functions that involve reasoning to help robots autonomously complete complex goals [101]. It refers to robots learning to complete tasks autonomously through their own interactions with the environment, such as obstacle avoidance, control, and various motion related tasks. This is an important part of robotics as cognitive reasoning allows robots to autonomously perceive and interact with real world environments. Where robots in factory lines are capable of doing complex tasks, most of them do not have perceptual abilities (beyond basic touch sensors); they are programmed to do one thing in a controlled environment.

Cognitive robotics is instead interested in robots that perceive, reason, remember, learn, and that can communicate with humans and each other [101]. The advantage this gives robotics is the ability to perform in a range of environments by adjusting their own behaviour through perceiving changes in the environment and reasoning about how to proceed [101]. Furthermore, complex tasks can be learned by the robotic platforms without the need for a human to develop a specific routine for each task [38]. This potentially allows for improved performance, as it avoids human bias, assumptions, and simplifications in conventional rigid models.

1.1 Motivations and Objectives

The broad objective of the work presented in this thesis is to apply human inspired mechanisms to address limitations in current robotic navigation techniques. In particular to the specific problems of ‘where am I?’ (SLAM) and ‘how do I get there?’ (path-planning). Two novel approaches inspired by how humans are considered to approach these problems will be presented in this thesis, one for each of the respective problems.

1.1.1 Cognitive action mapping

A number of SLAM techniques have been developed that effectively allow mobile robots to map unknown environments using a variety of range based or vision based sensors, see section 2.1. However, these approaches tend to rely on extracting high detail, feature rich maps of the environment. These maps are potentially storing large amounts of redundant or irrelevant information that could be filtered out, reducing computational complexity and memory requirements. Robots are also required to produce maps through their own independent sensor configurations and parameters, which may not translate to another robot without considerable human expertise being applied. Although mapping techniques such as grid and graph based maps are considered robot independent, varying placements of similar sensors can drastically change environmental perception [51]. Even the same model sensors can vary in their exact properties, requiring calibration before being used with many types of mapping approaches. In some cases comparing sensory perceptions between robots is infeasible. For example, a range based map is uninterpretable by a visual based robot platform and *vice versa*.

An ideal solution would only map salient information useful for navigation and in a format such that this information can be transferred between heterogeneous robots. Being able to transfer navigation information can improve co-operation between robotic platforms. Instead of having to

individually map an environment, once one robot has mapped a region it can share this information with other robots, decreasing the time required to explore and increasing the knowledge of the environment for all the robots involved. This can be useful in hazardous environments such as disaster relief, where low cost robots can be sent in to map the region before risking the higher cost robots designed for saving people.

Cognitive geography is the study of how humans perceive the world in their ‘mind’s eye’ [7]. Cognitive mapping is a process composed of a series of psychological transformations by which an individual acquires, codes, stores, recalls, and decodes information about the relative locations and attributes of phenomena in their everyday spatial environment [29]. A cognitive map is a spatial representation of the outside world that is kept within the mind until an actual manifestation (physically drawn map) of this perceived knowledge is generated, called the mental map [50]. In most cases, a cognitive map exists independently of a mental map. Cognitive mapping is the implicit (imagined), while mental mapping is the explicit (measured) part of the same process.

In practical terms the cognitive map is the abstract map or image a person has in their own head of an environment, this can be different for each individual. The mental map is the drawn map of the environment from the person, making the cognitive map into a physical description of the environment.

Cognitive maps are created through internal movement cues, and input from senses like vision, proprioception, olfaction, and hearing [50]. The perceived cognitive map consists of directional cues and positional landmarks. Directional cues can be both explicit cues, e.g. using a compass, or implied cues through internal movement relative to the environment. Positional landmarks provide information about the environment by comparing the relative position of specific objects, whereas directional cues give information about the shape of the environment itself. These aspects are then processed by the hippocampus together to provide a graph

(topological map) of the environment through relative locations.

Inspired by this perception of a separate cognitive and mental map in humans, a novel cognitive mapping stage will be presented that works in conjunction with current SLAM techniques, which are considered as the mental mapping stage. The mental mapping stage is specific to a robotic platform or sensor type, and will provide localisation for their respective robotic platforms. The cognitive mapping stage will seek to learn a high level action (physical movement cues) based map which is ubiquitous between robotic platforms. Similar to the human cognitive map, the robot cognitive map will be a topological map with regions within the environment linked by actions the robot can take. The main considerations of this technique are minimising the amount of unnecessary mapping information, reducing the memory costs of maps, and allowing shared navigation information through high level maps. The cognitive action mapping approach will thus be assessed on its ability to:

1. Reduce the amount of mapping information required for successful navigation.
2. Share cognitive maps between heterogeneous robotic platforms.
3. Share navigation information between visual and range base SLAM approaches.

Experiments will be conducted in a real-world environment with two heterogeneous robotic platforms with differing sensing capabilities.

1.1.2 Adaptive path-planning through emotions

Path-planning techniques are based on rigid computational models that rely solely upon the physical sensing capabilities of the robotic platform to determine a course of action. This makes path-planning approaches rigid

and inflexible to novel occurrences. That is, they can fail in new environments and cannot adapt to novel input or stimuli. Ideally a robotic navigation system should be able to adapt its path planning and behaviour to overcome a variety of obstacles within an environment without the need for specialised planning approaches. A specialised planning approach is one that is specifically designed for a particular environment, or has pre-programmed responses to states or objects within an environment. For example, detecting doorways or chairs is not generally applicable to outdoor environments. A complete rigid path-planning approach will have a complete mapping of all eventualities of these forms of states to responses for the robot to follow during navigation. Developing such a mapping is infeasible given the potentially infinite possible state-response pairs required to navigate the real-world.

What is required is a means of generalising the state-response pairs into a manageable set of patterns or behaviours, i.e. create categorised responses for a given set of states. However, determining the ideal patterns or behaviours by hand efficiently or effectively is not feasible given the large search space. Ideally, a robot should be capable of learning these beneficial behaviours from its own experiences in an environment without human intervention, avoiding human bias and providing the system the flexibility to adapt to novel environments.

Humans rely upon cognitive reasoning to interact with our environment, however our reasoning ability is influenced by our internal emotional state. Generally considered to be a detriment to a person's cognitive process, recent research is regarding them as a beneficial mechanism in the decision making process [23] [61]. Fellous [33] suggests that biological emotions facilitate the transfer of simple, but high-impact information, both externally (e.g. lack of energy) and internally (e.g. reacting to 'threatening' situations), for operating system-like tasks. Subsequently, recent years have seen an increase into research of robots and intelligent systems that possess emotion-inspired mechanisms. Generally targeted at social

robots for human interaction [9] [41], further work has also applied emotions to behavioural tasks such as navigation [58].

Inspired by these theories, a novel method will be developed to learn an emotion inspired model that can beneficially adapt a rigid cognitive path-planning approach based on the robot's interaction with the environment. Rather than mapping external stimuli directly to responses, the emotion model will seek to learn an intermediary set of emotion categories intended to achieve many of the same goals, but in a general way. An important function of emotions in the model is to provide a compact set of high-impact behaviour adaptations, reducing an otherwise tangled web of stimulus-response patterns into a more manageable structure.

In engineering and biology, the concept of a *bow-tie* is used to represent complex adaptive systems in a manner that provides flexibility without compromising efficiency of processing in the system. The shape of a bow-tie describes systems that include large numbers of inputs and outputs bridged by a smaller number of intermediary states and processes. Emotions may form the 'knots' of some of these bow-tie structures in biological cognitive systems, decoupling stimulus and response [65]. Manually pruning all possible connections to the bow-tie shape is an impractical task, with potential for added human bias. Therefore, the aim of this work is to use machine learning techniques as global search techniques to learn the bow-tie structures in a human-readable form.

From a purely algorithmic viewpoint, the robot will learn policies that adjust the parameters of its navigation algorithm in the form of bow-tie structures. From a physiological viewpoint, the behaviour of the robot based on its interaction with the environment will be used to determine the learned emotional response.

The emotion model will be assessed in two ways. The goal of the system is to learn a mapping of stimulus-response behaviours that beneficially adapt the robot's navigation behaviour. Based on this, the key assessment is the benefit of the system to the robot's navigation perfor-

mance through providing a generalised means of adapting its computational functions. This will be measured as the time to navigate the environment, and the number of collisions during navigation.

The secondary assessment of the learned emotion model will investigate the system for any emergent emotion responses. Emotions represented will not necessarily match their human counterparts, as the problem of mobile robot navigation differs from many problems that human emotions have evolved to address. Nonetheless, the results will provide an interesting insight into the formation of emotions. However, the primary aim of the system is to improve robotic navigation performance rather than attempting to imitate human emotion responses. Experiments will train the emotion system and compare the rigid computational model against the adaptive emotional system.

1.2 Thesis outline

Chapter Two An overview of robotic navigation techniques is provided, covering SLAM and path-planning approaches. Psychological perspectives on biological emotions is then presented with a review of related work using emotions in robotics. Machine learning techniques are the core mechanism for the approaches presented in this work, a review of machine learning is given with respect to robotics. Finally, Robot Operating System (ROS) is used as a means for controlling the robotic platforms in both simulation and the real-world, a description of ROS and the packages used in this thesis is provided.

Chapter Three Implementation details of the cognitive action mapping approach are provided. The decision points and common action features are described, and how machine learning is used to map them. The two robots utilised in this work and the environments used for testing are also introduced.

Chapter Four Quantitative experimental results of the cognitive action map being learned and shared between two real-world heterogeneous robotic platforms are presented and discussed. Including results demonstrating sharing information between visual and range based systems.

Chapter Five A description of the emotion model and how it adapts the cognitive navigation system is presented. Followed by details on how machine learning was used to learn the emotion model in both simulation and real-world environments.

Chapter Six Results demonstrating the utility of the learned emotion model against the computational system are given and discussed. Results of training the emotion model are also presented with any emergent behaviours in the system being examined.

Chapter Seven This thesis concludes with a summary of the contributions, the international publications resulting from this research, and a discussion of future work.

Chapter 2

Background

In order to achieve true autonomy in mobile robotics, a robot must be able to reason about its environment and follow a self determined course of action to complete given tasks without any human input. A major part of solving this problem is providing a means for robots to develop maps of their environment useful for navigation. Autonomous mapping of environments is covered by Simultaneous localization and mapping (SLAM) techniques, detailed in section 2.1. Autonomous navigation (detailed in section 2.2) falls under the domain of path planning relying upon SLAM to generate useful maps. This thesis introduces novel human inspired approaches to these problems to provide a means for a robot to learn from its interactions with the environment. Section 2.4 provides insights into the human emotion models that inspire these techniques. To implement these human inspired approaches, machine learning techniques will be utilised to provide a means for the robots to learn from their experiences. A review of machine learning techniques is given in section 2.5. The backbone of the approaches presented relies on Robot Operating System (ROS) for controlling the robotic platforms in both simulation and the real-world. A description of ROS and the packages used is provided in section 2.8. Finally, a summary of the background will be given in section 2.9.

2.1 SLAM

SLAM asks if it is possible for a mobile robot to incrementally build a consistent map of an unknown environment from an unknown location while simultaneously determining its location within this map. This can be considered a ‘chicken or the egg’ problem; to localise the robot requires a map but to create an accurate map the robot must be localised. Since robot motion is subject to error, the mapping problem induces a robot localisation problem. The basic principle of SLAM methods is to use environmental observations from a variety of sensors to help adjust for these movement errors, e.g. cameras, ultrasonic or LIDAR sensors. However, as sensors are not perfect, sensor noise introduces mapping errors into the system. The complexity of the technical processes such as locating and mapping under conditions of errors and noise do not allow for a coherent solution of both tasks. SLAM is a method that combines these problems into a loop with iterative feedback from one process to the other to enhance the results of both consecutive steps. Figure 2.1 shows the general process of SLAM algorithms. Different implementations of SLAM will substitute or add their own approaches to each stage in the system depending on the type of sensor the robot is using or environment it is operating in. For example a camera based approach could use SIFT features or line extraction for feature identification and comparison.

2.1.1 Statistical based SLAM

Smith and Cheeseman published the beginnings of SLAM techniques in 1986 [92]. This paper laid the groundwork for extracting a robot’s position, with an uncertainty, from observed and measured position estimates through an Extended Kalman Filter (EKF). Further work by Leonard and Durrant-Whyte then expanded the techniques to not only represent uncertainty, but to reduce or eliminate the uncertainty through observed measurements [60]. These early SLAM techniques were capable algorithms,

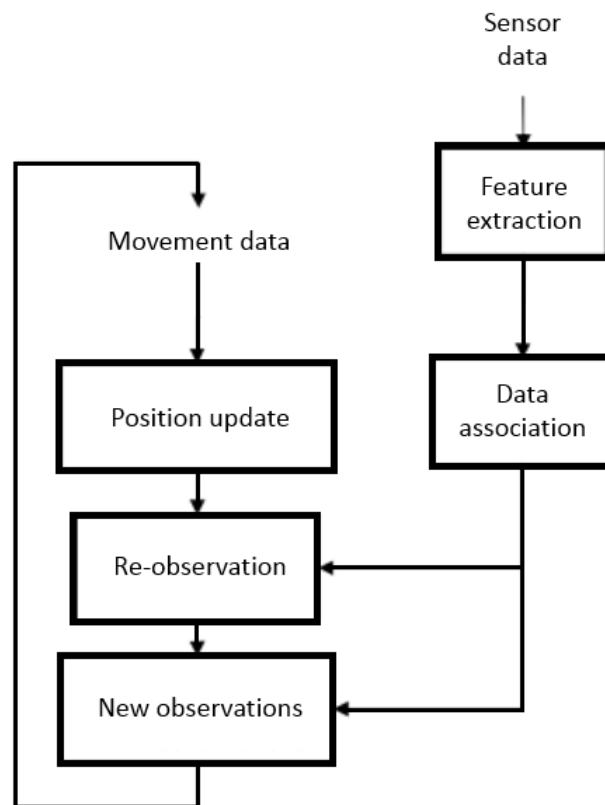


Figure 2.1: The position is estimated via internal motion readings. Landmarks are then extracted from the environment at the estimated position. Re-associated landmarks are then used to improve position estimate, while new landmarks are added to the map.

winning the DARPA Grand challenge in 2005 [100]. However, as Leonard and Durrant-Whyte [60] argued years earlier, these techniques still had fundamental issues in solving *data association*, *environmental dynamics*, and *computational complexity*. These three factors have driven development in SLAM until present day research, and are considered in the work presented in this thesis.

EKF SLAM [2] has issues with data association and computational complexity. Robots using these approaches rely upon range based sensors such as LIDARs and ultrasonics. These sensors require the landmark extraction stage to rely upon the position estimate of the robot, inherently adding uncertainty to landmark estimates. Over the course of environmental navigation this added uncertainty will eventually drive the position estimate into error, in which the robot cannot recover as it is unable to re-associate previously seen landmarks. EKF SLAM also has computational complexities in the calculation of the covariance matrix, with a quadratic scaling with each new landmark in the map. This prevents EKF SLAM from scaling to larger environments with tens of thousands of landmarks.

To tackle these problems Montemerlo et. al introduced the Rao-Blackwellized particle filter in FastSLAM [74] and FastSLAM 2.0 [73]. By having individual position estimates across the particle filter, FastSLAM is resilient to (but does not solve) data association errors from inherent uncertainty in landmark extraction. FastSLAM further improves upon EKF by utilizing a tree structure to efficiently represent landmark positions, reducing the computational complexity of landmarks to a logarithmic scaling. This allows FastSLAM to learn in environments with up to fifty thousand landmarks, which EKF approaches cannot computationally handle. Variants of FastSLAM are now considered to be the current de facto SLAM technique on range based sensor robots.

2.1.2 Appearance based SLAM

Range based sensors are computationally simple but financially expensive and limit SLAM approaches in data association and dynamic environmental problems. Vision based techniques are desirable as they are relatively cheap sensors which provide a rich range of information about an environment. Importantly, they also decouple the reliance on the robot's position estimate, with image comparison only requiring the images themselves. The limitation in early attempts was from inefficient and unreliable methods for image processing techniques in computer vision with early techniques relying on artificial landmarks for proof of concepts. Through advances in computer vision, visual based SLAM approaches have become a research focus.

Milford et al. introduced a rat inspired visual based approach, RatSLAM [71]. RatSLAM demonstrated a visual approach capable of resolving ambiguous landmark data even when subject to large uncertainty in position estimates. Since then vision based approaches have been scaled to large environments with up to 1000 km trajectories being mapped [21] [66]. Other notable visual based approaches are MonoSLAM [22], and FrameSLAM [55].

Vision based approaches all rely upon feature extraction techniques for precision in place recognition, with false positives being a major hindrance to localisation performance. Common feature finding approaches such as SIFT [63] and SURF [4] can handle scaling and rotations of images, enabling pose invariance. However, these approaches are unsuitable for dealing with perceptual changes in the environment such as day to night or sun to rain [76]. Condition invariance is required for solving long duration navigation based problems.

Several condition invariant solutions use training to dynamically model or predict changes in appearance [64] [95]. Training is also used to learn invariant place-dependant features [68] for place recognition. These techniques are limited by the computational overhead of collecting datasets,

and have limited applicability to unseen types of environments. Milford et al. introduced a new approach SeqSLAM [70], which relies upon matching a sequence of images over conventional approaches which make single place to place matches. SeqSLAM and its successor SMART [81] have been demonstrated to be robust for localisation through extreme changes in weather, addressing dynamic illumination in the environment. These two approaches are condition invariant, but have limited pose invariance. When simply changing lanes on the road, neither approach is able to successfully localise. Present work is attempting to bridge the gap between pose-invariant, condition-sensitive and the condition-invariant, but pose-sensitive place recognition algorithms [80].

2.1.3 Active SLAM

SLAM assumes an ideal goal directed path a robot can follow in order to explore the domain, which is likely not to exist in practical situations. Active SLAM approaches determine the robot's path based on reducing the SLAM uncertainty, improving mapping accuracy, and map completeness [32]. This is achieved via path planning that will select a path based on improving mapping accuracy and completeness, while also minimising the required time to explore. In contrast, the approach presented in this thesis seeks to learn a *map of actions*, in contrast to a sensor based map, that allow a robot to successfully traverse an environment. These actions are high level navigation information learned over time that can be shared between heterogeneous robotic platforms, as opposed to explicit low level navigation information aimed at improving the SLAM process.

2.2 Robotic navigation

Robotic navigation is the specific problem of finding a sequence of actions that will enable a robot to achieve a goal state given its current state [57].

For mobile robotics this involves safely and optimally navigating an environment. An ideal safe plan involves avoiding harm to the environment and the robot itself, either through preventing collisions or avoiding unsafe regions. An optimal or near-optimal path is considered with respect to time, distance driven, or energy used to reach the goal. Distance is most commonly considered. Accordingly, path-planning algorithms might calculate anything from a desired speed/direction of travel to an entire sequence of actions for the robot to follow.

Path-planning approaches generally rely on a map of the environment to generate these paths for the robot to follow. As discussed in section 2.1, localisation and mapping are interdependent processes as using a map to localise a robot requires that the map exists, while building a map requires the position to be estimated relative to the partial map learned so far. In contrast, path-planning is an separate process that takes place once the map has been built and the robot's position estimated. That is, first a map is generated from the environment, and then a path-planning approach utilises the map to generate a plan.

There are two common map representations used in path planning approaches, metric and topological maps [35] [69]. Metric maps represent the environment as a set of objects with co-ordinates in two-dimensional space. Topological maps represent the environment as place definitions and their relative positions are recorded as links. An example of the classic definitions of these representations is shown in figure 2.2.

The metric framework is the most commonly used representation by humans (world and road maps) and considers a two-dimensional space in which they places the objects in the environment. The objects are placed with precise and continuous coordinates. When displayed, a metric map generally looks like an architectural sketch and is easy to read for humans. This representation is very useful, but is sensitive to positional noise and it is difficult to calculate the object positions precisely from sensors due to uncertainties in their measurements.

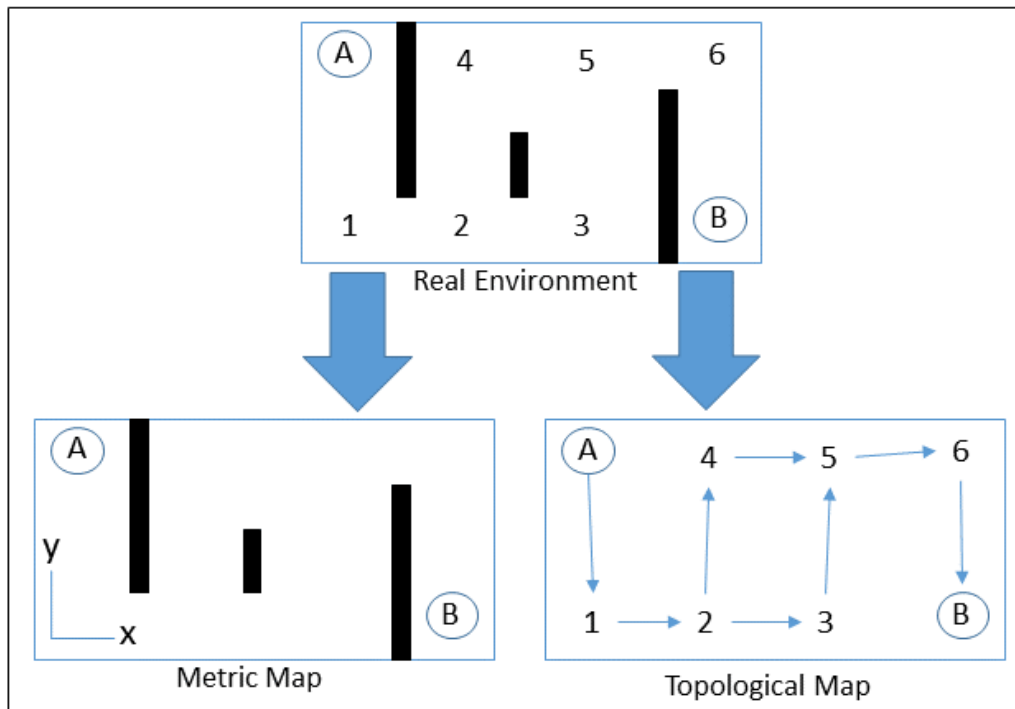


Figure 2.2: Illustration of the distinction between metric and topological maps. In the metric framework, object positions are inferred and represented in a common reference frame. Two positions, A and B, are represented in this map by their coordinates in this reference frame. These coordinates make it possible to infer their distance. In topological maps, places are stored with their spatial relations, the two positions A and B are recognized as individual places with links between them. This makes it possible to infer that position B can be reached from position A via the path 1, 2, 3, 5, 6 or 1, 2, 4, 5, 6.

The topological framework only considers places and relations between them, often being the distances between each place. The map is therefore a graph in which the nodes correspond to places and links correspond to the paths. In contrast to metric maps, topological maps only store locally relevant information. The downside is that learning and associating the nodes in the environment is a difficult problem. Either the nodes are too numerous and not locally unique, or are too few and thus hard to find.

Path-planning approaches are dependant on their mapping representations for creating paths. Metric frameworks provide a detailed map of the whole environment, allowing for detailed paths to be found at the cost of computational complexity. Topological maps inherently provide high level path planning through nodes in the graph and are computationally simple. But they do not allow for local control strategies as they lack details between nodes. A variety of path-planning techniques exist that utilise both or combinations of these representations.

Classical approaches to path-planning operate on graph search algorithms (such as A) and work well for planning an initial path through a known environment [34]. However, when operating in real-world environments robots do not have perfect information and graph based approaches may not be able to provide optimal or complete paths. An optimal path is defined on a heuristic and could be based on measures such as time, distance, or energy use. A complete path is a path that leads from the start to the end location with no missing steps.

One approach to improving the performance of graph based approaches is to re-plan the path each time new information becomes available. Re-planning each time new information is available is computationally expensive, especially in dynamic or large environments. Another option is to instead repair the original path to account for the changes in the map [93]. By repairing only the sections affected by the new information the system can improve the computational time but it will not guarantee an optimal path as it only adjusts the locally affected area [34].

Recent path-planning techniques utilise behaviour-based control methods to discover paths in the environment, e.g. state machines [42], [59]; biologically inspired systems based on artificial neural networks [42], [90]; fuzzy inference systems and/or evolutionary algorithms [52], [53], [79]; and search-based motion planners that employ detailed world representations [56].

These approaches are typically tuned, trained or evolved in a specific type of environment or a set of similar environments [77]. Once parameters are set, the system performs rigidly to the set parameters and take actions directly based on the external sensors of the robotic platform. That is, they can fail in new environments and cannot adapt to novel input that they have not been explicitly designed to handle *a priori*. To clarify, a path-planning approach may be able to adjust a plan to new information from the environment. However, it will not adjust its navigational behaviour and will not learn from its experience, in the same situation the approach will provide the same plan each time even if it leads to failure. This thesis is seeking to add a learning or memory mechanism through emotions to adapt a robot's navigation behaviour from its own experiences.

2.3 Bow-tie models

Learning a complete mapping of stimuli to response patterns is infeasible given the potentially infinite number of possible combinations a robot could require. Instead, it may be desirable to learn an intermediary number of states useful for filtering the environmental stimulus into manageable parts before determining a response. In engineering and biology these structures are commonly used and referred to as bow-ties.

Bow-tie architectures refer to ordered structures that often underlie complex technological or biological systems which are capable of balancing efficiency, robustness and evolvability in a natural setting [20]. Structurally bow-ties represent a system that maps many inputs through a few

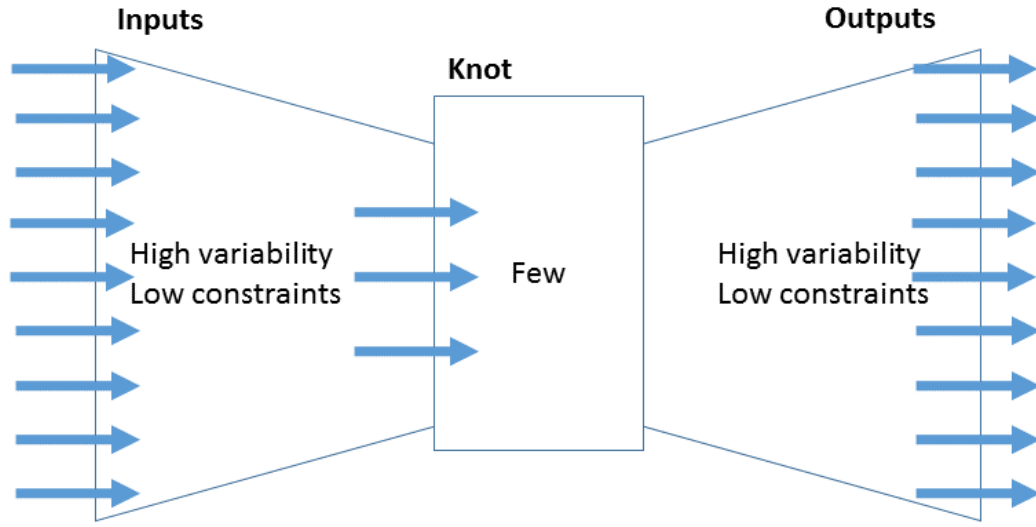


Figure 2.3: Many inputs with high variability are filtered through the 'knot' of the bow-tie. The knot then determines the output or response of the system.

intermediary states to many outputs. These architectures are used to manage a wide range of inputs through a core (knot), constituted by a limited number of elements, to a variety of outputs or responses. The knot acts as a manifold for the inputs to group or order the inputs into manageable parts. The knot then determines one of a possibly large variety of responses for the system to output. An example of this type of structure is shown in figure 2.3.

The emotion model presented in this thesis utilises the concept of bow-tie to learn a mapping of emotion based reinforcers (stimuli from the environment) to behavioural modifiers to adapt the behaviour of the robot based on its interaction with the environment. From a purely algorithmic viewpoint, the robot will learn policies that adjust the parameters of its navigation algorithm in the form of bow-tie structures. From a physiological viewpoint, the behaviour of the robot based on its interaction with the environment will be used to determine the learned emotional response.

2.4 Emotion models

Emotions are increasingly being regarded as a beneficial mechanism in the decision making process [23] [61]. Emotions, through an associative learning process, provide a low-level generalised behavioural response to a range of potentially beneficial or hazardous reinforcers from the environment [88]. Where a reinforcer is an emotional trigger associated to an aspect of the environment or sense of self. The low-level emotion response allows an organism to focus higher-level cognitive systems on important or specific aspects of the environment. Fellous [33] suggests that biological emotions facilitate the transferring of simplified, but high-impact information, both externally (e.g. lack of energy) and internally (e.g. reacting to 'threatening' situations), for operating system-like tasks.

Computational models of emotion can be broadly divided into those that define a small set of basic emotions (or emotion categories) such as anger, fear, joy and sadness, and those that model a set of dimensions such as valence and arousal, from which the various emotions can be inferred. Categorical models are often inspired by Ekman's cross-cultural research on human facial expressions [31], which suggests that humans universally recognise facial expressions of anger, fear, joy, sadness, surprise and disgust. Plutchik [83] argues that for an emotion to be regarded as basic or primary, it should also cross species boundaries. Plutchik's circumplex model [83] defines eight primary emotions, grouped into polar opposites (joy/sadness, anger/fear, anticipation/surprise, trust/disgust) and arranged in a 3D colour wheel, allowing adjacent emotions to be combined to form secondary emotions such as optimism (anticipation + joy).

Many robot implementations that attempt a complete, biologically plausible representation include Ekman's original six basic emotions. Other emotions are less frequently included, primarily because the various theories of biological emotion do not agree upon them (and perhaps also because they are applicable to a smaller set of task domains). Velsquez's

Cathexis architecture [99], one of the most influential early emotion-based robot controllers, models Ekman's six basic emotions. The robotic head Kismet [9], inspired by Plutchik et. al., expands this list to also include interest and boredom. Similarly, a typical set of emotions modelled in Moshkina and Arkin's TAME architecture [75] includes Ekman's six, and interest [1]. These approaches provide an insight into the use of emotions in robots for the goal of social interaction between humans and robots. The research in this thesis is seeking to utilise emotions for the benefit of robotic navigation.

Lee-Johnson and Carnegie have presented an emotion-modulated navigation architecture [58], which includes five of Ekman's six basic emotions, omitting disgust (which was considered to have limited usefulness in a robot that does not need to interact with humans or other robots), and adding interest (curiosity) and confusion. This work applies emotions to a hybrid reactive/deliberative planning and control architecture for a range of navigation and exploration tasks. Emotions are not modelled as discrete states or as internal sensors that drive action selection, but as continuous modulations of the robot's internal parameters throughout multiple computational layers. While many behaviour-based models regard emotions as potential replacements for deliberation, in their research they utilised emotions to augment the robot's deliberative capabilities by providing location-specific biases to path planning.

Each location specific bias was achieved by incorporating emotions into the map used for path-planning. Typically maps utilise sensor data to build a representation of the environment and path-planning will generate paths through the maps based on their path-planning heuristic. In the emotion model presented by Lee-Johnson the emotions are a further addition to the map that effect the generation of the path. For example, a sad emotion will cause the robot to move slower or cause the planning to avoid the particular area as if it was a potential hazard. In contrast to Lee-Johnson's work [58], the system presented in this thesis will not pre-define

how the emotions adapt behaviour and the assumption that the robot will have perfect information about its current state in the environment is removed [58]. The emotion responses will be learned by the robot from its own exploration of the environment.

Rolls [86] [87] presents a dimensional model in which emotions result from the presence, omission or termination of rewards and punishers, called instrumental reinforcers. Instrumental reinforcers can be primary, determined by genes (e.g. the tastes and smells associated with food), or secondary, learned by association with primary reinforcers (e.g. the sight of a favourite food). Figure 2.4 shows how specific emotions can arise from their respective reinforcement contingencies.

Primary instrumental reinforcers proposed by Rolls [86] include many that serve the goal of reproduction, such as courtship, parental attachment, nest-building and the crying of infants, which are not applicable to our robot (although some may be relevant to robots assigned the role of pet or caregiver). Reinforcers related to general social interaction, such as facial expressions, altruism, group acceptance and mind-reading (predicting others' behaviour), are likely applicable to multi-robot systems and human-machine interaction scenarios. Others related to bodily functions, such as tastes, odours, breathing, exercise and pain, are only relevant if some equivalent robot function is present (e.g. an analogy can be drawn between food or sleep, and a mobile robot's need to recharge its batteries). The reinforcers that are most likely to be relevant for our tasks are those that can be applied to the general adaptive behaviour of individuals, including novel stimuli, habitat preference, control over actions, problem-solving, and play.

A combination of factors such as the reinforcement contingency (whether a reward or punishment is presented, withheld or terminated), the intensity, the primary and secondary reinforcers involved, and whether an active or passive response is possible, can account for a wide variety of emotions. The presentation of a reward might correspond with satisfac-

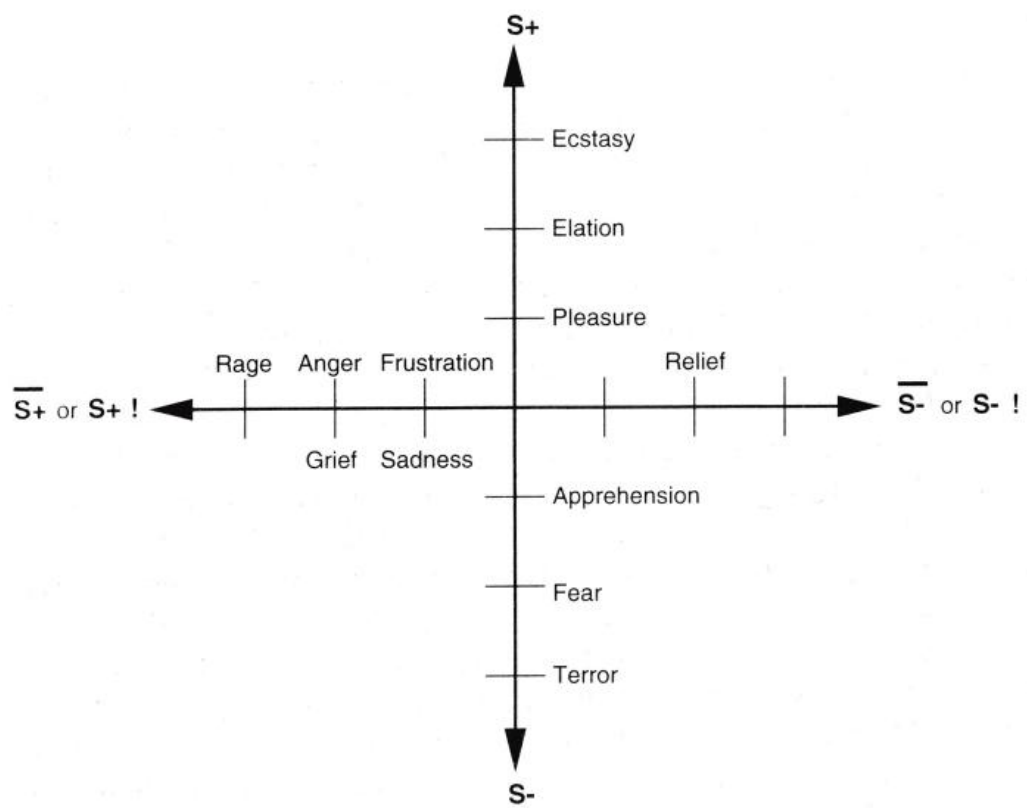


Figure 2.4: Framework for understanding specific emotions as arising from specific reinforcement contingencies. From "Emotion Explained" [86]

tion or happiness, while its omission might result in frustration or anger if an active response is possible. If the reward is terminated, or if no active response is possible, an animal may instead experience sadness or grief. Similarly, the presentation of a punishment might result in fear, or, if the punishment is inescapable or recurring, depression, whereas its omission or termination are likely to elicit relief. For example, Ishii et. al. [49] demonstrated that rats tormented by a robotic “bully rat” over an extended period of time showed signs of depression (characterised by immobility).

Rolls [86] is not a proponent of categorical studies of emotion such as those conducted by Ekman [31], arguing that some of those approaches risk finding a small number of emotion categories, even if that results in an incomplete model, because “seven, plus or minus two” is the maximum number of categories with which humans normally operate [72]. However, in the robotics domain, a higher level of completeness might be considered excessive and impractical. It can be argued that there need not be a perfect correlation between human, animal and computational emotions, given their vastly different cognitive architectures. A robot need not include a complete set of biological emotions if a smaller subset can provide sufficient adaptive capabilities for the tasks it must perform. Such tasks might include avoiding hazards that the primary sensors cannot easily detect (e.g. glass walls), or responding quickly and appropriately to unexpected or dynamic situations.

In a sense the robotic emotions in this thesis can be considered as a low level response to the environment based on internal stimuli. Where path-planning traditionally uses external senses to control the behaviour of the robot, the emotional reinforcers are an internal sense of the robot’s current performance. The term emotion in this sense is a control policy. A control policy that learns to adapt the robot’s navigation behaviour based on internal stimuli.

The model for path planning presented in this thesis is inspired by

these theories. Rather than mapping stimuli directly to actions, or learning the actions in a single step, the robots learn an intermediary set of emotion categories intended to achieve many of the same goals, but in a more general way. An important function of emotions in the model is to provide a compact set of high-impact behaviour adaptations, reducing an otherwise tangled web of stimulus-response patterns into a more manageable structure.

Emotions have been applied to path-planning before by Johnson et. al, showing an emotion model that beneficially adapted a robot's path-planning [58]. However, this work assumed perfect localisation and had pre-defined emotion responses. Perfect localisation means there is no uncertainty in the position estimate, which is unrealistic in robotics. Pre-defined emotions means the system had human encoded emotion response for environmental conditions, creating potential human bias to the navigation behaviour. The model presented here will learn these behavioural responses and have no pre-defined emotion states.

2.5 Machine learning

The previous sections have detailed the limitations in current SLAM and path planning techniques. Machine learning techniques provide methods that can potentially allow a robot to autonomously learn to solve these navigation problems through its own interactions with the environment. Machine learning techniques have been used successfully in many robotic problems, making use of the abilities of the techniques to adapt to their problem environment [24] [25] [27]. This section will provide an overview of Machine learning and the techniques used in the work presented in this thesis.

Machine learning seeks to develop methods of allowing computers or robots to learn to act without being explicitly programmed to. The advantage of learning allows a system to be adaptable, organise new knowledge

into effective general representations, and discover new facts through observation [113]. This allows a robot to respond to and learn from novel input from the environment. Machine learning techniques learn through systematic trial and error approaches, relying on feedback from the environment to determine the quality of their actions [30]. Depending on the type of feedback, machine learning algorithms can be classified into three main categories: supervised, unsupervised and reinforcement learning.

Supervised Learning The agent is learning with labelled class examples or instances. The desired outputs for a problem are known in advance, and the goal is to learn a function that maps inputs to desired outputs.

Unsupervised Learning The agent is learning without labelled class examples, which means there are no correct answers for the agent to explicitly learn from. It attempts to find inherent patterns that can then be used to determine clusters for given instances.

Reinforcement Learning Desired outputs are not directly provided. Every action of the agent has some impact to the environment, and the environment provides feedback on the quality of its action in the form of scalar rewards and punishments. The agent learns based on the rewards and punishments it receives from the environment.

Supervised learning is generally used in classification problems, learning models to match data to given labels. Unsupervised learning is used for clustering problems, learning to find labels for data. Reinforcement learning (RL) is useful for the domain of robotic navigation, providing a means for a robot to learn from interaction with its environment.

2.5.1 Reinforcement learning

A large variety of problems in robotics can be treated as Reinforcement Learning (RL) problems [54]. RL is closely related to the theory of classical

optimal control, a familiar approach in the field of robotics [84]. Both RL and optimal control seek to address the problem of finding an optimal policy (controller or control policy) that optimises an objective function (the accumulated cost or reward). Both also rely on the notion of a system being described by an underlying set of states, and there being a model that describes transitions between these states.

Optimal control assumes perfect knowledge of the system's description in the form of a model, i.e., a function that describes what the next state of the robot will be given the current state and action. For such models, optimal control ensures strong guarantees which often break down due to model and computational approximations [54]. In contrast, RL operates directly on observed data and rewards from interaction with the environment, allowing RL techniques to handle problems which are analytically intractable using approximations and data-driven techniques.

For example the common multiplexer problem attempts to create a map of the inputs to a multiplexer to its output [45]. Even a relatively simple multiplexer at 11 bits has $2^{11} = 2048$ possible combinations to manually check, increasing this to 135 bits becomes impractical to check every possible combination of inputs to outputs as the computational power required does not exist to provide the answer in a timely manner. Using LCS Iqbal et. al have shown that is possible to learn a set of rules that can map the inputs to output (without having to check every possible combination) through RL [45]. A concise discussion of viewing reinforcement learning as 'adaptive optimal control' is presented in [96].

In standard RL models a learning agent has a perception of its state in the environment and a set of possible actions the agent can effect. These actions effect a change in the environment, changing the current state of the agent, which produces a scalar reinforcement signal (reward). In many cases, the reward is arranged by the experimenter or trainer of the technique. For instance, in a classification context, the reward may be 1.0 for 'correct' and 0.0 for 'incorrect'. In a robotic context, an example of reward

could be a number representing the change in distance to a recharging source, with more desirable changes (getting closer) represented by larger positive numbers. Reward is used by the system to alter the likelihood of taking an action given what has been learned previously in similar conditions. The agent learns to map these state and action combinations to their utility (ability to provide reward), with the aim of the learning to maximise reward from the environment. The learning system learns this mapping over time by systematic trial and error, relying on a variety of search techniques for guidance.

2.5.2 Challenges of RL in robotics

Although suited for robotics, RL is generally a hard problem, and many of its challenges are most apparent in robotic applications. These limitations must be taken into account when choosing appropriate machine learning techniques.

As the number of dimensions in the search space grows, exponentially more data and computation are needed to cover the complete state-action space, coined the curse of dimensionality by Bellman [5]. Robotic systems often have to deal with high dimensional states and actions, notably in anthropomorphic robots. In robotic navigation, the action space is continuous in the directions and behaviours a robot may choose, creating a very large search space. The chosen RL technique will need to be efficient at searching in order to feasibly learn, both in a time and learn-ability perspective, on a robotic platform. Learn-ability is whether or not the learning system can learn a solution in the given problem domain.

Learning the state and action mapping is not a trivial task as reward signals in robotics are generally sparse and unspecific. Sparse means the rewards are not always available and may only occur after a long sequence of actions have been carried out. Unspecific means the rewards provide little information as to how well the robot has performed, often being just a binary reward value for succeed or fail. Meaningful differentiation be-

tween different solutions is non-trivial, two solutions may succeed but no further information as to quality can be determined from the reward.

Robotics deals with real-world systems where sensor and environmental noise add uncertainty to observations. The RL technique will not be able to precisely know the state of the robot and will have to handle noise in the data. This makes learning state action mappings difficult as two similar states may require different actions to be considered optimal and the system will be required to reliably differentiate between them. Further challenges are discussed by Kober et al. [54], providing an in-depth survey of using machine learning techniques in real-world and simulated robotics.

The aim of this work is to apply human inspired systems to robotic navigation. As described above, to learn in a real-world environment a machine learning technique requires: an ability to generalise states to reduce memory and computation, the ability to handle sparse and unspecific reward, and the ability to handle noisy real-world input.

Learning Classifier Systems (LCSs) are desirable for this task as they can generalise rules to cover large search spaces, are designed to operate in RL problems, and can learn with noisy real-world data. A further advantage of LCSs is they produce human readable solutions in the form of “IF condition THEN action” rules. This transparency is in contrast to other machine learning approaches such as neural nets which work as a black box and require extensive examination to interpret the results. As LCS allows us to examine the results and interpret what the system has learned.

2.5.3 Learning Classifier Systems

A Learning Classifier System (LCS) is a machine learning technique that learns a population of “IF condition THEN action” rules called classifiers [11]. A classifier’s condition is a representation of a state within a problem environment. These learned rules provide a means for the system to

choose the best action given its current state in the environment. Evolutionary Computation (EC) techniques and heuristics are used to search the space of possible rules, whilst RL is used to assign utility to the rules.

EC techniques are machine learning algorithms based on Darwin's principles of evolution. EC uses the concepts of natural selection and genetics to evolve (learn) a population of optimal solutions to a given problem. Central to the idea of EC is the idea of searching a problem space by evolving an initially random population of solutions such that fitter (higher quality or more optimal) solutions are bred (generated) over time. This is seen as the population being adapted to the problem environment; fit solutions survive and breed to produce fitter solutions. For more on EC a detailed introduction and history is given by Eiben and Smith in [30].

LCSs were first introduced by Holland based around his seminal algorithm the Genetic Algorithm (GA) [38] with influences from Q-learning [102]. LCS aimed to provide a technique capable of credit assignment under conditions of sparse reinforcement, distributed and generalisable representations of complex categories, and adapting system knowledge through interaction with the environment.

This initial LCS was considered to be overly complex and practical experience did not demonstrate the desired learning behaviour/performance [112]. Wilson then introduced the "zeroth-level" LCS (ZCS), which simplified the original framework of Holland's LCS and improved performance [110], becoming the new standard framework for future LCSs shown in figure 2.5. ZCS is a strength based classifier system, which defines classifier fitness as the amount of reward expected if the agent takes a particular action in the given state. ZCS's representation of fitness is limited in that it only searches niche regions of the state space that provide reward, leaving low reward, no reward, or even negative reward regions un-searched.

To address this Wilson introduced an accuracy based LCS called XCS with a new fitness definition [111]. XCS defines fitness as the accuracy of the classifier to predict the expected reward for an associated action. XCS

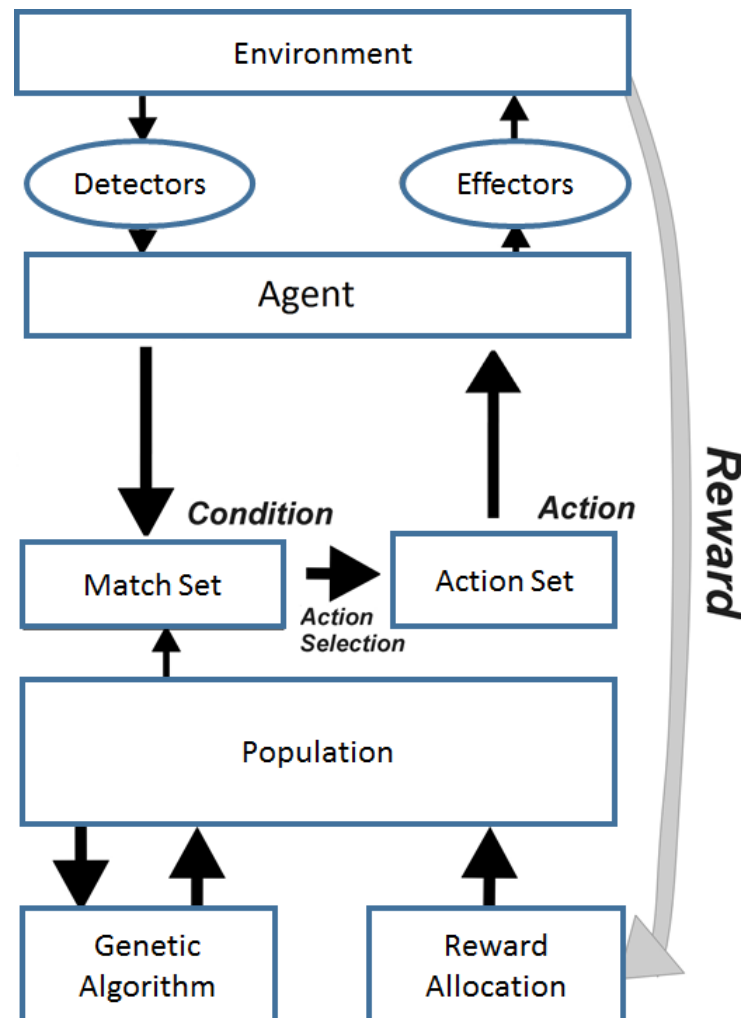


Figure 2.5: An agent uses sensors to detect the current environment state, finds the rules in the population that match the current state, and selects from them which action to take. The agent then affects the action on the environment, receiving a reward signal which is allocated to the classifiers. Finally, a Genetic Algorithm is used to search for new and potentially useful classifiers.

attempts to maximize the accuracy of the predicted reward value from taking a given action. This definition of accuracy based fitness allows the evolution of classifiers that are both maximally general and accurate in their prediction of payoff [111]. This allows classifiers to learn accurate rules that predict all reward conditions, covering the complete search space.

The standard State-Action classifier has no memory mechanism and can only learn optimal rule sets in Markov environments, where the best action is determined solely by the state of the current sensory input. Real-world environments are only partially observable Markov environments from an agent's sensory information. Within these environments, Condition-Action classifiers either fail, or only develop a suboptimal policy due to aliasing of the rules [116].

To compensate for this lack of memory, Anticipatory Classifier Systems (ACSs) have been developed that remember previous states [94]. The ACS is a learning algorithm that combines the LCS with the learning theory of 'Anticipatory Behavioral Control' [118]. The insight from psychology is that higher animals form an internal environmental representation and adapt their behaviour by forming anticipations [13]. The knowledge of an ACS is represented by Condition-Action-Expectation rules.

Traditional LCS are limited as they are unable to apply previously learned domain knowledge from smaller problems to more complex or larger problems in the same domain. This requires LCSs, as in the case of many other approaches, to re-learn from the start for each new problem. To address this, recent work by Iqbal et. al. [44] has incorporated genetic programming based representations to action selection, XCS using Code Fragmented Conditions XCSCFC. This code fragmented representation allows a LCS to build on previously learned information and apply it to more complex problems without the need to re-learn. This reduces the number of training instances required in large problems, but requires extra time due to more involved methods [43].

LCSs initially used binary values in the state to represent the problem

environment. Research has since expanded the binary rule representation to include real-valued state representations capable of representing real-world inputs [12]. The LCS also has a 'don't care' operator it can assign to any elements within a state/condition. The 'don't care' operator, usually denoted as #, acts as a wild card value that will match any input to that element in the state. This allows for generalisation of rules. As a binary example, a classifier with the state 1#1 will match inputs 111 and 101. A generalised rule in the case of mobile robotics will have only the relevant sensor values required to define the state, for example filtering out redundant or irrelevant sensor data from the robot's sensors. Generalised rules compact the overall number of rules required to solve a problem. This reduces the overall memory and computational requirements for a mobile robot.

Fuzzy Classifier Systems have also been developed for handling continuous real-world inputs through fuzzy rule-based models [6]. These fuzzy rules follow the form of "IF x is A THEN y is B ", in contrast to the traditional structure of classifier rules "IF condition THEN action". These fuzzy rules have been used primarily for classification problems with real-world continuous data [48] [115]. Fuzzy classifiers have also been used in robotic control problems [82], however for this work the traditional rule set of learning actions given conditions is considered to have a direct relation to the decisions a robot must make while navigating.

In summary, the LCS is an efficient global search technique that is commonly used to find transparent (human readable) State-Action or State-Action-Expectation [117] rule sets. This thesis extends these representations in order to utilise the capabilities of the LCS. The next two sections will provide details on the operation of the XCS and ACS algorithms used in this thesis.

2.6 XCS: Accuracy based LCS

The overall goal of XCS is to form a complete and accurate mapping of the problem space (rather than simply focusing on the higher payoff niches in the environment) through efficient generalisations. In RL terms, XCS learns a value function over the complete state-action space. In this way, XCS represents a way of using traditional RL on complex problems where the number of possible state-action combinations is very large. The output of XCS is a population of rules to allow an agent to make the best action in any given state within the environment. The best action is considered the action that will provide the most reward from the environment to the agent. Provided in this section is a general overview of the XCS algorithm, a full description of the XCS algorithm can be found in [17].

2.6.1 XCS classifier

XCS learns a population of classifiers which represent its knowledge about the problem environment. Each classifier is a Condition-Action rule with attributes:

Condition Specifies the input states (sensory state of the environment) in which the classifier can be applied.

Action Specifies the action that the classifier proposes the agent enacts, e.g. the direction the robot moves.

Prediction Estimates the reward the agent will receive for enacting the action if the state matches the classifier's condition.

Fitness A measure of the classifier's accuracy of its prediction; the higher the value the more accurate the classifier is.

Numerosity Reflects the number of micro-classifiers (ordinary classifiers) this macro-classifier represents. Classifiers in XCS are macro-classifiers,

i.e. each classifier represents n micro-classifiers having identical conditions and actions.

The dot notation will be used to refer to an attribute of a classifier cl , i.e. $cl.C$, $cl.a$, $cl.p$, $cl.F$, $cl.n$ respectively. There are also three notable sets used throughout the XCS algorithm.

Population $[P]$ consists of all classifiers that currently exist in XCS.

Match $[M]$ consists of the classifiers in $[P]$ that match the current state S .

Action $[A]$ consists of the classifiers in $[M]$ that propose the enacted action.

2.6.2 Learning parameters

XCS is an online learner that takes in a new input from the environment and determines the best action to take in that state. The best action is the action that is expected to give the highest reward from the environment. XCS learns this by iterating through a trial and error process until an end condition is met. The end condition can be either a pre-determined number of iterations or until a given classifier fitness level is reached.

Each learning iteration, a new state S is generated from the environment through the learning agent's sensors. XCS will then generate $[M]$ from the classifiers that match S with their condition C . Covering is then called if the size of $[M]$ is below a set threshold. Covering is used to generate classifiers to match a new S , in particular at the beginning of a training run when the population is empty. A new classifier is created to match the current S , and is associated with a random action to propose (as there is no information available to choose an action). The new classifier is then added to $[M]$. For each possible action a in $[M]$, the system prediction $P(S, a)$ is computed as the fitness weighted average of the rules that advocate a in $[M]$

$$P(S, a) = \frac{\sum_{cl.a=a \wedge cl \in [M]} cl.p * cl.F}{\sum_{cl.a=a \wedge cl \in [M]} cl.F} \quad (2.1)$$

giving a prediction of reward for each possible action from the current state.

XCS will then select an action a from those advocated in $[M]$ and create $[A]$. This can be done in two different ways: exploitation, select the highest predicted performing rule; or exploration, randomly select a rule in $[M]$. Exploration is used to prevent locally optimum rules being ingrained by exploring random areas of the search space. During training the selected action alternates between exploration and exploitation. The selected action is then effected in the environment by the agent, a scalar reward r is returned to the system prior to the next input state S_{t+1} based on the outcome on the environment. If exploitation is used then a GA will be applied to $[A]$ in order to generate potentially better rules.

The GA is used as the primary discovery method for new, potentially promising, classifiers. A stochastic search method is necessary as the continuous domain means an exhaustive search is impractical and a discretised domain needs human expertise and/or generates imprecise rules. The GA selects two parents from $[A]$ depending on the classifiers' fitness, e.g. through roulette selection or tournament selection. The two parents are copied then crossed over and mutated to produce two new children [38]. Each child is then checked for subsumption, a test that determines if a parent is an accurate and more general form (covers more problem instances) of its child. If the child is not subsumed it is inserted into the population, else the parents' numerosity is increased by one as it covers the new micro classifier. If the population is full, another classifier will be deleted based on the XCS deletion method [15].

Reward is provided by the environment when a goal location or condition is discovered, sometimes requiring multiple actions to be enacted. If no reward signal is received (different from a reward signal of zero) then XCS will iterate over the steps described above until a reward signal is

given. When a reward signal is received, XCS will update the relevant classifier parameters.

2.6.3 Parameter update

XCS updates the fitness of the classifiers in the action set $[A]$ based on the reward received from the environment. First the predicted reward $cl.p$ for each classifier in $[A]$ is updated by the actual reward r with a learning rate β ($0 \leq \beta \leq 1$), typically 0.1.

$$p \leftarrow p + \beta(r - p) \quad (2.2)$$

In the same way the prediction error ϵ is updated.

$$\epsilon \leftarrow \epsilon + \beta(|r - p| - \epsilon) \quad (2.3)$$

Next, the fitness is updated in three stages. First, the absolute accuracy κ of a classifier is derived from the reward prediction error ϵ ,

$$\kappa \leftarrow \begin{cases} 1 & \epsilon < \epsilon_o \\ (\epsilon/\epsilon_o)^\nu & otherwise \end{cases} \quad (2.4)$$

where ϵ_o controls the maximal tolerance for prediction error ϵ , with ν a constant controlling the rate of decline in accuracy κ when ϵ_o is exceeded. Essentially any classifier with ϵ below ϵ_o is considered accurate, while ν controls the difference between degrees of non-accurate classifiers.

Second, the relative accuracy κ' is derived from the absolute accuracy κ , which assists in allocating rules evenly throughout the search space.

$$\kappa' \leftarrow \frac{\kappa}{\sum_{[A]} \kappa} \quad (2.5)$$

Finally the fitness of each classifier F is updated based on the relative accuracy.

$$F \leftarrow F + \beta(\kappa' - F) \quad (2.6)$$

In the case where multiple actions have been taken to obtain reward, the update method is run for each action set from the start of the learning iteration to the final iteration which received the reward. The reward is propagated down the action sets, usually with a discount factor or problem specific approach. This is similar to Q-learning [102], where an agent take numerous actions before receiving reward from the environment.

XCS does not have a memory mechanism. XCS can thus only learn an optimal policy in Markovian environments where in every situation, the optimal action is always determined solely by the state of current sensory inputs. But in many applications, the agent has only partial information about the current state of the environment, so that it does not know the state of the whole world from the state of the sensory input alone. The agent is then said to suffer from the hidden state problem or the perceptual aliasing problem, while the environment is said to be partially observable with respect to the agent.

In non-Markovian environments a robot knowing only its current state may not be able to know (sense) its global position. In order to generate a useful action mapping the system will require a memory mechanism to remember previous locations and the actions the robot must effect to navigate between states. ACS provides a suitable memory mechanism and is described in the next section.

2.7 ACS: Anticipatory classifier system

ACS has the same output as XCS, a population of rules to allow an agent to make the best action in any given state within the environment. However, ACS also provides the expected effect, next state, from affecting a given action on the environment. With this rule set, ACS has been trained with robots in simulated mazes [118] and with robots in controlled small scale

environments [14] with good results being shown. Provided in this section is a general overview of the ACS algorithm, a full description of the ACS algorithm can be found in [16].

2.7.1 ACS classifier

ACS extends the XCS's Condition-Action classifier to incorporate expectations through a Condition-Action-Expectation classifier representation. ACS can then differentiate between aliasing states by learning to expect what the next state will be for taking a given action. Each classifier is a Condition-Action-Expectation rule with attributes:

Condition Specifies the input states in which the classifier can be applied.

Action Specifies the action that the classifier proposes the agent enacts, e.g. the direction the robot moves.

Expectation Specifies the expected next state of the agent if it enacts the proposed action.

Prediction Estimates the reward the agent will receive for enacting the action if the state matches the classifier's condition.

Fitness Is the measure of the classifier's expectation accuracy; the higher fitness the more likely the agent will be in the predicted next state, therefore, the more useful the classifier is for path planning.

The dot notation will be used to refer to an attribute of a classifier cl , i.e. $cl.C$, $cl.a$, $cl.E$, $cl.p$, $cl.F$ respectively.

2.7.2 Learning parameters

The ACS algorithm operates in the same manner as XCS, whereby an agent perceives a state within the environment and effects an associated

action following the same process as described in section 2.6.1. This process is repeated until the environment provides a reward signal for reaching the goal state. ACS differs from XCS in how the reward signal is processed, and adds an additional reward signal for correct expectations.

In ACS an action is always accompanied by an expectation of its result on the environment, the expected next state S_{t+1} in the case of robotic navigation. By comparing the differences of the perceived result from acting on an environment and a classifier's expectations, ACS learns a population of Condition-Action-Expectation classifiers [94]. For example if a robot moves in a given direction (effects an action) it will learn to expect where it will be within the environment after completing the action. A robot can then traverse these expectations to find a path to a desired goal state.

Reinforcement is provided to ACS in two ways: from the environment, when the robot discovers the goal state; and internally when a classifier successfully anticipates the new state, arriving where the robot predicted. When a reward signal is received, ACS will update the relevant classifier parameters.

2.7.3 Parameter update

ACS performs the same as XCS with the addition of the expectation update. When the environment provides a reward signal to the agent, the classifier's reward prediction is updated following the XCS update method shown in section 2.6.1.

ACS updates the fitness of each classifier when the expectation of a classifier in $[A]$ matches the next state correctly. Therefore, classifiers with a high fitness accurately predict the next state (after taking a given action) and are therefore useful for path planning. Classifiers with high reward prediction are useful for navigating to specifically learned goal locations as it has been learned that their actions reliably lead to the goal location, and subsequently provide reward. There are three cases where a classifier's

fitness is updated:

Useless case The useless case is when no change in perception is perceived from the environment after taking a given action. In this case the expected fitness q of each classifier in $[A]$ is decreased: $q = q - b_q * q$ where b_q is the learning rate for the expectation.

Unexpected case The unexpected case is when the new state does not match the expected prediction of the classifier in $[A]$. In this case a new classifier will be generated that matches the incorrect classifier's $C - A$ but with the current state as the E . The incorrect classifier is then penalized as in the useless case.

Expected case The expected case is when the new state does match the expected state of the classifier in $[A]$. In this case the expected fitness of the classifier q is increased: $q = q + b_q * q$, where b_q is the learning rate for the expectation.

2.8 Robot operating system

Robot Operating System (ROS¹) is a open source software framework for robot software development, providing operating system like functionality on a heterogeneous computer cluster [85]. ROS provides standard operating system services such as hardware abstraction, low-level device control, implementation of commonly-used functionality, message passing between processes, and package management. It is based on a graph architecture where processing takes place in nodes that may receive, post and multiplex sensor, control, state, planning, actuator and other messages. This makes ROS a powerful research tool in robotics as software can be developed for robotic platforms almost regardless of the physical hardware and sensors.

¹<http://www.ros.org/about-ros/>

The work presented in this thesis will utilise ROS (distribution Hydro Medusa) as the core system for SLAM, navigation, and communicating between the robot and the machine learning techniques. GMapping will be used for range based robots, while ROS RatSLAM will be used for vision based robots. The navigation stack will be the base navigation system that the emotion model will adapt.

The ROS package GMapping² is a state-of-the-art implementation of a Rao-Blackwellized particle filter [28]. GMapping successively estimates the pose and the map based on sensor readings and generates a metric two-dimensional occupancy grid map. ROS RatSLAM [3] is an open source implementation of RatSLAM designed to operate within the ROS framework. Based on continuous attractor network dynamics, RatSLAM is capable of mapping by closing loops to correct odometry error. In conjunction with a camera, RatSLAM produces an estimate of the robot's position. However, RatSLAM does not produce a standard sensor map of the environment, but rather a Cartesian map of the robot's path within the environment. Technical details on Open RatSLAM can be found in [3].

The navigation stack³ [67] is a commonly used robotic navigation algorithm that provides path planning and navigation controls for generic robotic platforms, with a variety of parameters that require tuning for optimal navigation for each platform. The navigation stack utilises a metric grid based map of the environment for path planning, created using the robot's external sensors and localisation through SLAM algorithms. The emotion system will learn to adjust these parameters to control the behaviour of the robot's navigation.

Machine learning algorithms can take a large number of iterations to learn a solution, this can take from hours to days, up to even months, depending on the problem. Running real-world experiments over these times frames is not always feasible, with battery time, access to testing

²<http://wiki.ros.org/gmapping>

³http://wiki.ros.org/base_local_planner

environments, and health safety requirements to consider, i.e. requiring active personnel to prevent damage to property or persons. This necessitates that longer experiments are run in simulation where these concerns are not a problem. Gazebo⁴ is an open source extensive robotics simulation tool, which provides physics simulations of robots in various environments, and is extensively used in projects such as the DARPA robotics challenge. Gazebo offers the ability to accurately and efficiently simulate robots in complex indoor and outdoor environments, providing the ability to rapidly test algorithms, and train them in realistic scenarios.

2.9 Summary

The current state of robotic navigation has been discussed in this chapter, covering both SLAM and path-planning techniques and their respective limitations. A perspective on human emotions and their possible application to robotic navigation has then been discussed. Finally, machine learning (in particular LCSs) have been discussed as a means of implementing the human inspired robotic navigation presented in this thesis.

The two novel human inspired approaches are presented in the next four chapters, the first addresses limitations in SLAM, while the second addresses limitations in path-planning. The next chapter presents the implementation of the cognitive mapping approach called Cognitive-Action-Mapping SLAM (CAM-SLAM). Inspired by the concept of human cognitive maps, CAM-SLAM is seeking to map only the regions in which a robot must make decisions on its action within the environment. Unlike previous navigation systems where sensor data is considered salient, this new approach focuses on these decision points as the primary features of the map. The goal of these features is to reduce the amount of mapping information required for navigation, and to provide a common reference frame for heterogeneous robotic platforms to share navigating information. This

⁴<http://gazebosim.org/>

technique is then trained on real-world robotic platforms, with chapter 4 presenting and evaluating the results.

Chapter 5 presents an emotion inspired path-planning technique. Current path-planning is limited by their rigidity, emotions provide a means of adapting a robot's behaviour based on its interaction with the environment. The emotion system will adapt a computational approach's parameters (ROS navigation stack) based on the robot's internal emotion states, which are influenced by the robot's interaction with the environment. In contrast to Lee-Johnson's work [58], this system will not define how the emotions adapt behaviour and the robot will not have perfect information about its current state in the environment [58]. The emotion responses will be learned by the robot from its own trial and error. The results of the emotion model will be discussed in chapter 6, providing results into the performance benefits, if any, to the robot's performance. This is then followed by discussions on the emergent behaviour from the learned emotion model, what behaviours has the robot learned, and how they might be considered as representations of human emotions.

Chapter 3

Cognitive Action Mapping SLAM

Ideally SLAM algorithms would only map the most salient landmarks of the environment for navigation and in a format such that mapping and/or navigation information can be shared between heterogeneous robotic platforms. Note that sharing navigation information (what action to take given a situation) is crucially different to sharing map information (given situations only). The ability to share navigation information between heterogeneous robots would be advantageous for completing common goals such as mapping, goal locating, and path planning.

Currently SLAM approaches rely on extracting high detail maps of the environment, building feature rich maps from range sensors or storing thousands of images for place recognition. These maps are potentially storing large amounts of redundant or irrelevant information that could be filtered out, reducing computational complexity and memory requirements which are limited on small scale robot platforms. Robots are also required to produce maps through their own independent sensor configurations and parameters, which may not translate to another robot without considerable human expertise being applied. Although mapping techniques such as grid and graph based maps are considered robot independent, varying placements of similar sensors can drastically change environmental perception [51]. Even the same model sensors can vary in their

exact properties, requiring calibration before being used with many types of mapping approaches. In some cases comparing sensory perceptions between robots is infeasible. For example, a range based map is uninterpretable by a visual based robot platform and *vice versa*.

Inspired by the perception of a separate cognitive and mental map in humans, this chapter presents a novel cognitive mapping stage to the SLAM process which seeks to address these issues. The cognitive mapping stage will work in conjunction with current SLAM techniques, which will be considered to be the mental mapping stage. The mental stage will learn a detailed sensory map of the environment, as per normal, providing localisation for the robot platform. The cognitive mapping stage will seek to map a series of novel “decision points” with associated novel “action” features linking them together. Decision points are constructed on the physical actions a robot can take in a given region, where the actions are physical directions the robot can move. These high level decision points are similar to a topological based map and should reduce the information required for navigation. This technique being presented is called Cognitive Action Mapping SLAM (CAM-SLAM¹).

An advantage of using physical actions for mapping is that they are ubiquitous across robotic platforms, even with varying sensing abilities. This allows the cognitive mapping stage to be independent of the mental mapping stage, only requiring a position estimate from SLAM. Therefore, the common action features should be able to be utilised to provide a means for heterogeneous robotic platforms to share navigation information. The robots are effectively providing information about the environment through directions and landmark queues based on the choices they should make about their direction of travel at certain points. Colloquially, this is similar to providing directions to the pub for a friend “take the first left, the second right, and then head straight until you see it”. The indi-

¹formerly Learned Action SLAM (LA-SLAM), however changed to keep terminology in-line with physiological inspirations.

vidual robot will be required to track its location through its own means, while using the directions (cognitive map) of the co-operating robot to guide it.

However, determining which decision points are most salient during mapping is a non-trivial problem, being computationally impractical to consider all possible combinations of features in a continuous real-world domain [62]. It is hypothesised that through the generalization abilities of the LCS, a robot may learn to only consider the salient decision points within the environment such that it can successfully navigate, while filtering irrelevant or redundant aspects, e.g. excessive landmarks or poor paths.

3.1 CAM-SLAM model

CAM-SLAM seeks to learn a map of decision points (see section 3.1.1) with associated actions (see section 3.2) for the robot to traverse between them, termed the Cognitive Action Map (CAM). CAM-SLAM autonomously detects and learns these decision points as the robot traverses the environment through the machine learning approach ACS, which represents the decision points as Condition-Action-Expectation classifiers. A visual representation of the CAM-SLAM approach is shown in figure 3.1.

3.1.1 Decision points

Decision points are regions within an environment where the robot must make a decision on the action it will perform, for example an intersection in a corridor. In robotic navigation the action is the direction the robot will move in an attempt to find a goal location, whether to turn left or right at the intersection. These key points are similar to landmark features, however they are prominent for the action decisions the robot can make as opposed to sensory information in standard landmark association.

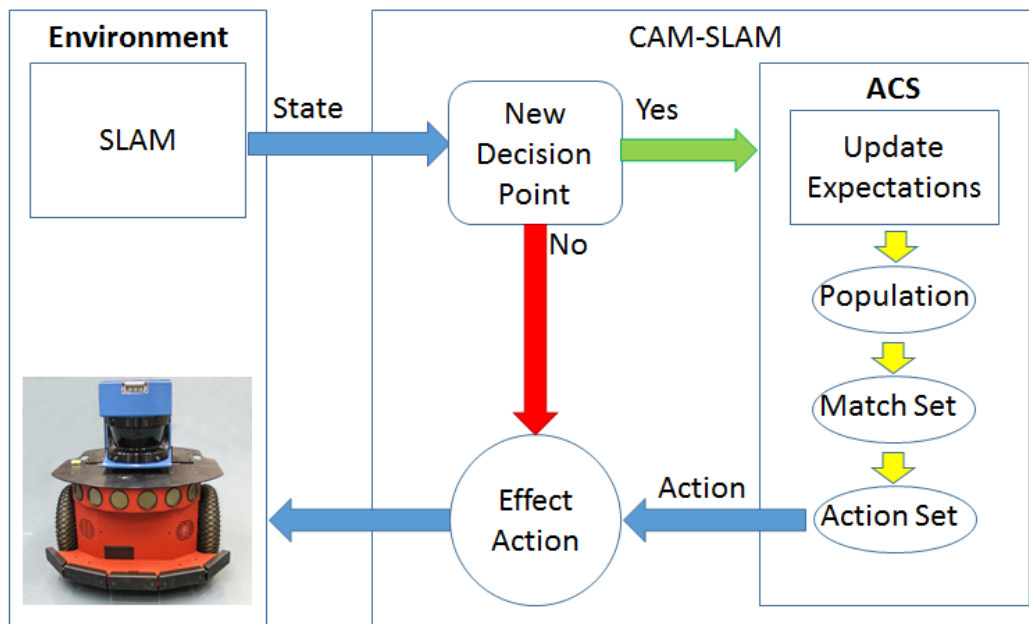


Figure 3.1: Flow diagram of the proposed CAM-SLAM. The robot navigates within the environment using SLAM to track a position estimate. ACS uses this position estimate and the robot's sensor data to learn the high level CAM of the environment.

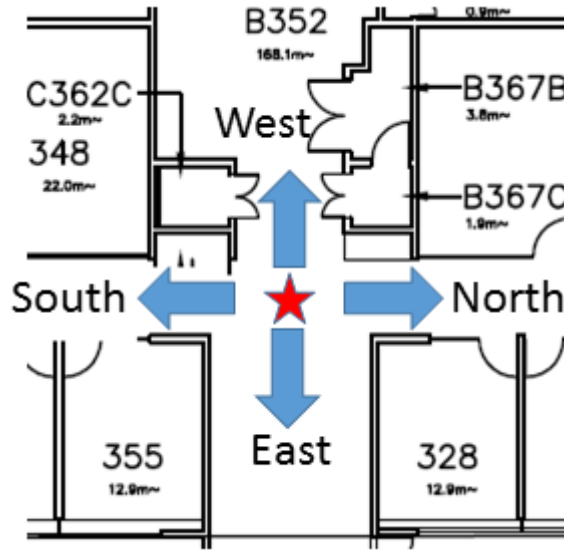


Figure 3.2: An example of a decision point, with the possible actions a robot can take. Arrows indicate the possible actions and the red star shows the location of the decision point. Note that decision points are not known *a priori* so the system must learn their optimum placement, which may be any location in the continuous domain

Decision points contain only the locally relevant position and sensory estimates required to associate the decision point with a robot's sensory configuration. Each decision point is then linked via an action association. These action associations predict the next decision point the robot should reach if it were to follow a given action. This creates a graph like structure of decision points in which a robot can navigate.

The robot autonomously determines decision points through its current state, sensor conditions, and the actions it determines are available. A robot will recognise a new decision point if a new action is detected, or the current action has become infeasible (e.g. there is an obstruction in the way). Once a new decision point has been recognised, the robot will stop and determine a new action to perform. An example of a decision point is given in figure 3.2.


3.1.2 Mental mapping: SLAM

CAM-SLAM utilises SLAM for localisation only relying on the CAM to provide path planning through decision points. The abstraction of the SLAM algorithm provides the ability to utilise the SLAM approach best suited to the given robot and its given sensor configuration, while still being able to utilise the navigation information from other robot platforms. For example, a vision based robot is best suited for vision based SLAM approaches, and range based robots are best suited for statistical based approaches. The two SLAM approaches used as mental maps in this work are GMapping for range based robots, and RatSLAM for vision based robots, both of which are described in section 2.8.

3.2 Anticipatory Classifier System

An ACS is used to learn a population $[P]$ of Condition-Action-Expectation ($C - A - E$) classifiers. These classifiers represent a physical location in the environment with a position estimate and associated sensor data. A classifier example from CAM-SLAM is shown in a classifier form in table 3.1 and in a map form in figure 3.3. Table 3.1 shows that the sensor data is stored as the condition of the classifier and the associations between decision points (other classifiers) are stored as expectations. Classifier one is from a range based sensor, where the condition data is distance measurement between 0-360 degrees of the robot. Classifier two is from a camera, the condition data is then an image at the location of the specific decision point. Figure 3.3 shows classifier one in standard mapping format within the CAM. These autonomously learned $C - A - E$ classifier populations $[P]$ thus represent the CAMs and can be learned through ACS.

Table 3.1: Example of the environmental state as a classifier condition. Classifier one is an example of a classifier learned by a range based robot, and classifier two is an example from a vision based robot

ID	Condition	Expectation (Action → Classifier)			
		North	East	South	West
1	2, 3, 4...20, #, 30, #, 33	2	N/A	N/A	N/A
2		2	N/A	N/A	N/A

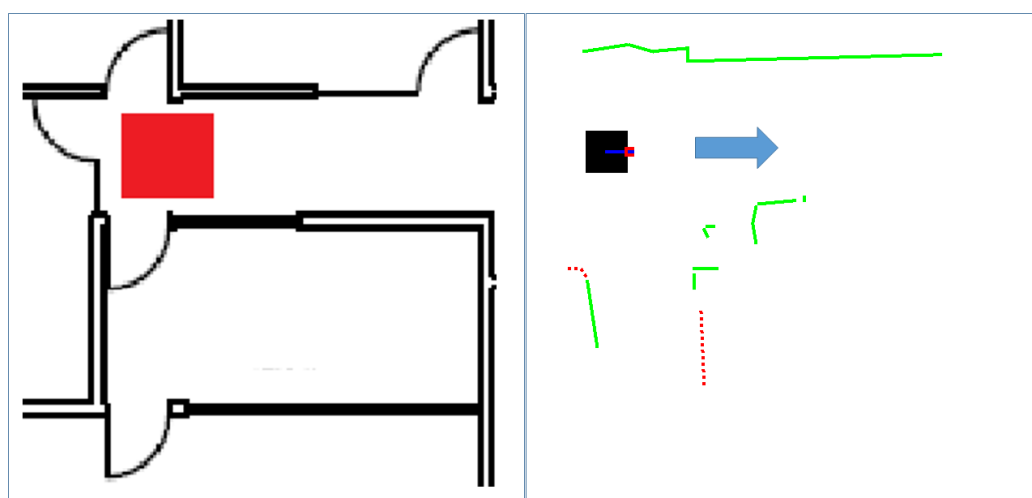


Figure 3.3: Left: Ground truth matching the classifier's position. Right: Classifier example: The black square represents the robot's estimated position when the classifier was generated, the blue line indicates orientation and the smaller red square the LIDAR position. Lines extracted from LIDAR data for classifier matching are shown as green lines, with the red dashed lines indicating lines set as 'don't care'.

Condition

The Condition is the state, position and sensor estimates of the decision points within the environment. As the state values are estimates from the real world they will contain uncertainties due to noise. The Condition mitigates the noise through an upper and lower bound around each value determined via the associated condition's uncertainty. The position estimate is taken from the robot's SLAM approach, with the upper and lower bounds set by the uncertainty (variance) in the position estimate computed by the SLAM algorithm. The sensor estimates are obtained from the sensors on the robot platform, with the upper and lower bounds set by the respective sensor's physical measurement uncertainty. For example, if the distance measurement is 0.90 m and the sensor's uncertainty is ± 0.03 m then the upper and lower bounds will be 0.93 and 0.87 respectively. This allows for cases where the robot is in the same place but due to noise the sensor is not reading the same measurement.

Action

In the real-world the number of possible actions a robot can perform is potentially infinite, with all possible directions being available with a high enough precision in movement. To reduce this action space, and for ease of testing validity, the number of possible actions is discretised into a set of four possible directions the robot can choose to move, shown in figure 3.2 on page 53. Increasing the number of actions increases the number of expectation mappings the ACS is required to learn, which extends the required number of training iterations to a point where this becomes impractical in real-world robotics. A discrete action set is a viable compromise for testing the validity of the approach and is a reasonable assumption for the testing environment (see section 3.4.2) where the need for a continuous action set is minimal. A robot determines which actions are feasible through its sensor input, for example the LIDAR can determine

feasible actions based on the distance the robot can travel in a given trajectory. In the case of the vision sensor, this work will utilise a sonar sensor in the same manner as the LIDAR to determine feasible actions, however only the vision sensor will be used for localisation estimates.

Expectation

The Expectation links the classifiers through associating actions. Each classifier predicts which classifier, and hence which decision point, the robot will reach next if it takes a given action. These associations provide path planning information, with the robot able to follow a series of actions between decision points.

3.3 Implementation

As has been described above the CAM is represented by a population $[P]$ of *Condition – Action – Expectation* classifiers. In order to learn these classifiers the ACS approach described in section 2.7 will be used. To train the ACS the robot is tasked to find a given location within the environment, with reward being provided for successfully finding it. Each learning iteration is a single traversal from the start position to the goal location, with numerous iterations required to learn the $[P]$ (CAM). The robot has no initial knowledge of the goal state's position; however the robot can recognise the goal independent of its position estimate through external sensors, for example a camera to recognise a red sign. This has the consequence that the robot can find the goal location but may not be estimating its own position correctly. The ACS will need to learn with potentially erroneous inputs if SLAM position estimates fail.

During navigation, when the robot determines it has reached a decision point, the robot will stop and send its current state S to the ACS. The match set $[M]$ is then formed of all classifiers in the population that match S . To determine if S is a match to a classifier cl the condition for each is

compared in three stages:

Sensor matching

In Sensor Matching the sensor data in S is compared to the sensor data in cl . Focusing on matching the sensor data helps to generalise the classifier conditions to contain only sensor data specific to a single state within the environment. Classifiers with overly general sensor data will suffer from aliasing and incorrectly match various states. These classifiers will have inconsistent expectation predictions eventually leading to removal from P . Conversely classifiers that are consistently accurate in their predictions will increase in fitness.

Sensor matching can utilise a variety of methods for different environments. However, as sensor matching is not the main focus of this research, simple but effective solutions in the testing environments were used:

LIDAR Matching LIDAR matching compares the 2D point cloud data from the LIDAR in S and a classifier's LIDAR condition cl_{LIDAR} . Each point in the cloud is stored as a Cartesian point with an origin of zero. Each LIDAR point in S is associated to its closest Euclidean match in cl_{LIDAR} . A pair of points is considered to be the same if the Euclidean distance is less than the uncertainty in the LIDAR measurements. A matching percentage M_p is calculated based on the total number of matches over the total size of the smaller LIDAR data.

$$M_p = Matches(S_{LIDAR}, cl_{LIDAR}) / Min(S_{LIDAR}, cl_{LIDAR}) \quad (3.1)$$

If M_p is over 95%, the classifier is considered a match to S and is placed into $[M]$. The 95% threshold was determined from empirical testing in the environment. Ground truth LIDAR data was taken at different locations in the environment. The robot was then placed at each location to tune the matching threshold. The threshold with the

highest accuracy of correctly matching the locations to the ground truth was selected. This quick and simple procedure will be required for each new domain type, future work will seek to replace this approach with a pose and condition invariant solution.

Different LIDAR sensors have varying capabilities (e.g. range or resolution), and can be mounted at various positions on a robot; making a direct comparison of their outputs non-trivial for complex environments. For the structured domain used in these trials, this approach is reasonable and is used for simplicity for initial verification of CAM-SLAM. This approach may be extended to more complex domains through various feature extraction techniques.

Image Matching Image matching compares the image captured in S with a classifier's image condition in cl_{Image} . This process follows the same approach utilised by SeqSLAM [70]. Both images are down sampled from their original size to a set size (48x16 pixel resolution) and converted to grey-scale before processing. Each image is then put through patch normalization to reduce local variation, which has been shown to enable robust scene recognition in varying illuminations [119]. The Sum of Absolute Differences (SAD) between both images is calculated as:

$$SAD = \frac{1}{S} \sum_{x=0} \sum_{y=0} |p_{x,y}^i - p_{x,y}^j| \quad (3.2)$$

If the SAD score is less than 0.10, determined in the same process as the LIDAR threshold, the classifier is considered a match to S and is placed into $[M]$.

Action matching

Action Matching compares the position estimates and the possible actions that S and a cl 's condition perceive to be feasible. In the case that sensor

types are different, this process allows a mode of comparison through the common action set. In the case the sensors are the same but no match is found, this process provides a secondary means of comparison if the sensor's configurations are sufficiently different to cause matching failures.

If the position estimates are within a given distance (1.0 metre plus the variance of the measurement) then sensor data will be compared based on the perceived available actions from the common action set. For example, if each sensor's data determines that the robot can only take actions *north* and *south* then the condition is considered a match. If both conditions are met cl is added to $[M]$.

If Action matching is used, a new classifier cl_n will be created from S in an attempt to adjust to the new perspective in the environment. cl_n will initialise with half the average quality rating of the classifiers that match S to propagate previously learned information. It is unknown whether the new classifier is useful or not, to propagate more fitness may hinder the system if it is poor but to propagate less fitness if it is good will also hinder the system. Propagating half the fitness is a common compromise in this situation. This is to help seed the learning and improve the re-learning time in the new area. cl_n is added to both $[P]$ and $[M]$.

Position matching

If both the Sensor Matching and Action Matching fail to find a match, the system relies on the position estimates. This allows for cases where the robot has a high confidence in its current position estimate but the environment has changed, either a new action has become available or an existing action is no longer feasible. If both the cl 's and S 's position estimates are confident (variance less than 0.2 m) and the Euclidean distance is less than 1.0 m, then they will be considered a match.

This third matching case will create a new classifier cl_n from S , in an attempt to adjust to any potential change in the environment. cl_n will carry over half the average quality rating of the classifiers that match S

to propagate previously learned information. However, cl_n is added to both $[P]$ and $[M]$ if and only if $[M]$ is empty after checking all classifiers in $[P]$ against S . This is to mitigate creating classifiers when the robot's position estimate is in reality in error.

3.3.1 Action selection

Action selection will alternate between; exploitation, select an action a with the highest payoff prediction; or exploration, randomly select an a in $[M]$. This is to prevent locally optimum paths being ingrained. The action set $[A]$ is generated from the classifiers in $[M]$ that advocate a . Finally, a GA is applied to $[A]$ to generalise the sensor data and reduce redundant information.

3.3.2 Parameter update

Classifier parameters are updated in two cases. First, the fitness of the classifier is updated depending on the difference between the expected and the actual result of taking a given action. Second, the reward prediction of the classifier is updated when it receives a reward from the environment, i.e. finds the goal location. Each of the rewards is a static value and does not diminish over time, with the robot receiving the full reward each time it successfully finds the goal location or correctly predicts the next location.

Expectation update

Expectation update learns the action associations between classifiers (decision points) to provide path planning information. Classifiers in A are rewarded based on their predictions of the next state. The classifiers in A will predict which classifier (decision point) will be encountered from the robot taking a .

Useless case No change in perception is perceived from the environment after taking a given action, for example the robot does not move. In this case the fitness q of each classifier in $[A]$ is decreased: $q = q - b_q * q$ where b_q is the learning rate for the expectation.

Unexpected case The new state does not match the expected prediction of the classifier in $[A]$, the robot encounters a different decision point. In this case a new classifier will be generated that matches the incorrect classifier's $C - A$ but with the new state as the expectation. The incorrect classifier is then penalized as in the useless case.

Expected case The new state does match the expected prediction of the classifier in $[A]$. In this case the quality of the classifier q is increased: $q = q + b_q * q$.

The fitness of a classifier is an indicator of how reliable a classifier is at predicting the next state and how useful it is for path planning. The higher the fitness of the classifier, the more reliable the classifier is for path planning.

Action update

Action update rewards the classifiers that successfully lead to the given goal location. As the robot will enact multiple actions before reaching the goal, the reward must be distributed to all classifiers involved. The reward is distributed with a discount factor to all classifiers in all $[A]_i$ that have been used to get to the goal location.

$$reward_i = Reward_{total} * discountFactor^{-i} \quad (3.3)$$

Each classifier's reward prediction $cl.r$ is then updated with the discounted reward value $reward_i$.

$$cl.r \leftarrow cl.r + \beta * (reward_i - cl.r) \quad (3.4)$$

The last action effected will receive the full reward, as it brought the robot to the goal location, while earlier actions will still receive some reward for contributing. The disadvantage to this approach is that the discount factor needs to be tuned to the length of the chain of actions, in this case it was set to 0.5 from empirical testing. An algorithmic description of CAM-SLAM is given in algorithm 1. The next section will provide details on the experimental set-up used to train CAM-SLAM.

3.4 Experimental setup

CAM-SLAM will be validated by training CAM-SLAM on two robotic platforms with different morphologies. Training of the robots will be conducted in the office environment of the Cotton building at Victoria University of Wellington (VUW).

3.4.1 Robotic platforms

Ownbot

“Ownbot” was designed and built for autonomous navigation of indoor environments at VUW, see figure 3.4 on page 66. The platform utilises a Hokuyo LIDAR URG-04LX-UG01 placed 0.14 m off the ground for landmark extraction with a maximum range of 5.6 m. Mecanum wheels give the platform omni-directional movement, with Hall effect sensors giving odometry estimates with a precision of 0.6 mm per tick. With no IMU for corrections, Ownbot does suffer an average Euclidean error of ± 50 mm over one metre. The consequence of this is that SLAM has to account for a relatively large uncertainty in movement. Ownbot is equipped with a Fit-PC 3 Pro, 1.65 GHz dual core, 4 GB DDR3 RAM, running Ubuntu 12.04 with ROS Hydro. An Arduino Mega 2560 controls the motor drivers and communicates with the on-board computer via `roserial`².

²<http://wiki.ros.org/roserial>

Algorithm 1: CAM-SLAM algorithmic description

Data: $[P]$: population of rules; cl : classifier $\in [P]$; cl_n : new classifier;
 $[M]$: match set; M_p : Match set position; $[A]$: action set; $[A_s]$:
action set stack; a : action; S : state;

```

1   $[A_s] = []$ ;
2  repeat
3       $S = \text{robot.processNewState}();$   $\backslash\backslash$  Robot reads new pose and
        sensor state
4       $[M] = []$ ;  $[M_p] = []$ ;
5       $\backslash\backslash$  Find classifier in  $[P]$  that match the current state to create  $[M]$ 
6      foreach  $cl$  in  $[P]$  do
7          if  $\text{matches}(S_{\text{sensor}}, cl_{\text{sensor}})$  then
8               $[M].\text{add}(cl)$ ;
9          else if  $\text{matches}(S_{\text{position}}, cl_{\text{position}})$  then
10              $[M_p].\text{add}(cl)$ ;
11      $\backslash\backslash$  If  $[M]$  is empty cover new state and utilise classifier which
        match the position estimate
12     if  $[M].\text{isEmpty}()$  then
13          $[M] = [M_p]$ ;
14          $cl_n = \text{cover}(S, [P])$ ;
15          $[M].\text{add}(cl_n)$ ;
16     if  $[M].\text{size}() \leq 4$  then
17          $cl_n = \text{cover}(S, [P])$ ;
18          $[M].\text{add}(cl_n)$ ;
19      $\backslash\backslash$  Update the classifiers in the previous match sets expectations
20     if  $![M]_{-1}.\text{isEmpty}()$  then
21          $\text{updateFitness}([M]_{-1}, S)$ ;
22      $M_{-1} = [M]$ ;
23      $\backslash\backslash$  If the goal is found update classifiers reward prediction, else
        determine an action and enact it on the environment
24     if Goal Discovered then
25          $\text{rewardActions}([A_s])$ ;
26     else
27          $[A] = \text{createActionSet}([M])$ ;
28          $\text{applyGA}([A])$ ;
29          $[A_s].\text{put}([A])$ ;
30          $\text{robot.followAction}([A])$ ;
31 until Goal Discovered;

```

Pioneer

The Pioneer P3-DX³, see Figure 3.4, is designed and built by Adept for robotic research. The Pioneer P3-DX is a differential drive robot platform with configurable sensing capabilities and is popular in mobile robot research. The Pioneer is equipped with a Mamba Dual-Core Computer, 2.26 GHz dual core, 8GB DDR3 RAM, running Ubuntu 12.04 with ROS Hydro. Where Ownbot uses smaller, lower accuracy sensing equipment, the Pioneer utilises higher quality accurate sensing equipment. The Pioneer utilises gyro corrections with its odometry-based position estimate. This gives the Pioneer a greater accuracy in its movement estimates, resulting in an average Euclidean error of ± 5 mm over one metre compared with ± 50 mm for Ownbot. The main range based sensor for the Pioneer is a SICK LIDAR LMS-100 placed 0.40 m off the ground. The LMS-100 can give range estimates up to 20 m across 270 degrees at a resolution of 30 mm. This gives the Pioneer a greater mapping size and accuracy than Ownbot at 5.6 m and resolution of 3%. Finally, the Pioneer has been equipped with a Kinect sensor providing 3D range data and vision data, however only the vision data is used in this work.

3.4.2 Environment

CAM-SLAM trials were run on the second and third floors of the Cotton building at VUW, see figures 3.5 on page 68 and 3.6 on page 69 respectively. These environments have a common office layout, which have been used to test a variety of different SLAM techniques. As per standard offices, the area provides a range of features and obstacles, such as corridors, various chairs and tables, glass walls, doorways, and rubbish bins. The Cotton building is also a convenient location for testing this research as it is in the same building as this work was developed. The testing areas are small compared with state-of-the-art SLAM algorithms, where the size

³<http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx>



Figure 3.4: Left Pioneer: Adept robotic research platform. Right Ownbot: designed and built at Victoria University of Wellington for indoor autonomous robotic navigation.

of the domain is restricted to provide practical learning times for ACS. No *a priori* information was provided to the learning algorithm.

Ground truth for each floor was determined based on the action set available for the robots, e.g. north and south. Each point in the environment at which the robot will be required to make a decision about its current course of action is considered a decision point, e.g. intersections and T-junctions in the corridors. Based on this criteria ground truth has been determined and shown for each floor in figures 3.5 and 3.6 on page 69.

3.5 Experiments

The first set of experiments aims to validate CAM-SLAM by training the system individually on each robot platform. The second experiment will benchmark the memory requirements of CAM-SLAM against a standard SLAM algorithm. The third test will determine how CAM-SLAM handles changes in the environment. The final two experiments will test CAM-SLAM's ability to share navigation information between two heterogeneous robotic platforms.

3.5.1 Experiment one: Learning a CAM

The first autonomous learning experiment is to validate the system on the two heterogeneous robotic platforms. This will provide a comparison on the performance of CAM-SLAM on robots with varying capabilities, and robots utilising different SLAM approaches. First, the robot Ownbot will be trained on the third floor of Cotton, shown in figure 3.6 on page 69, using its LIDAR and GMapping. Second, the Pioneer robot will be trained on the second floor of Cotton, shown in figure 3.5, first with its LIDAR and then with its camera (via the Kinect), using GMapping and RatSLAM respectively. The shared learning experiment will then swap each robot and provide the respective CAMs of each floor learned by the other robot.

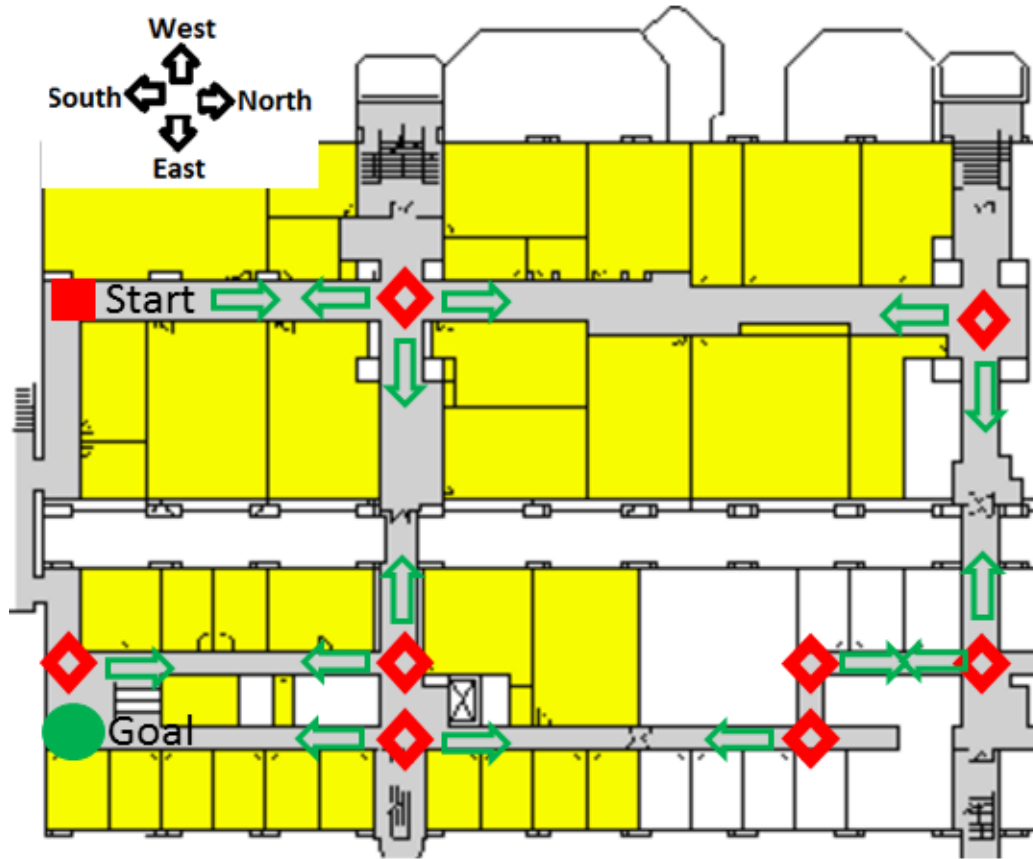


Figure 3.5: Ground truth layout of level two of the Cotton building, at Victoria University of Wellington, used for real-world experimentation. Red diamonds: human ground truth decision points to assess the learning of the robot. Green Arrows: potential actions the robot can take. An action is taken until the robot autonomously determines it has reached a new decision point, which are unknown *a priori*, without access to the ground truth. The domain is continuous such that the robot must determine its own decision points, which will be compared to the ground truth to judge performance. Where **North** is faces to the right of the image.

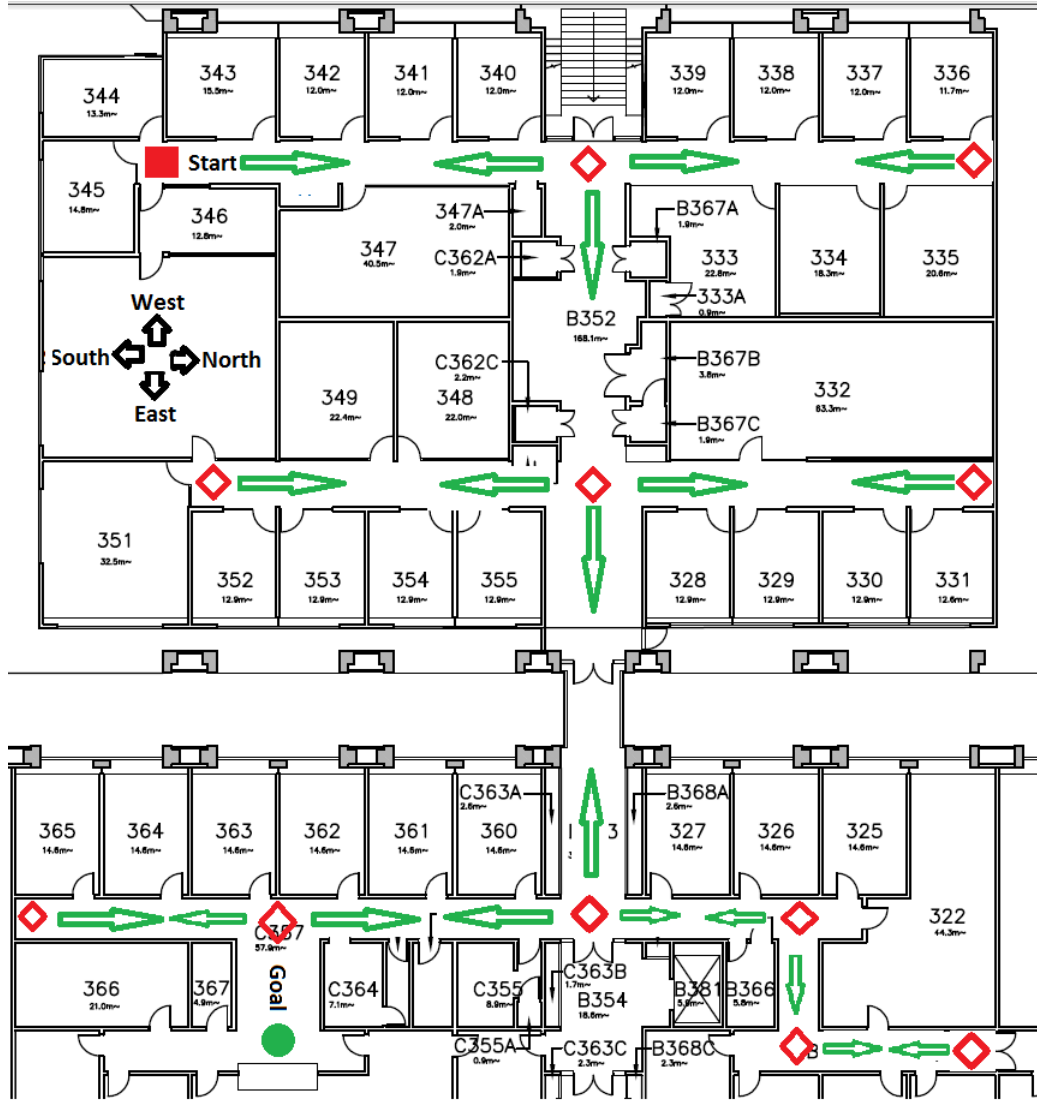


Figure 3.6: Ground truth layout of level three of the Cotton building, at Victoria University of Wellington, used for real-world experimentation. Red diamonds: human ground truth decision points used to assess robot performance only. Green Arrows: potential actions the robot can take. Where **North** is faces to the right of the image.

3.5.2 Experiment two: Benchmark algorithm

As no other algorithm, known to the author, seeks to share maps with path planning information embedded, no direct comparison can be made. However, a comparison with human in-the-loop mapping to autonomous path planning can be conducted. Ownbot will be manually driven through Cotton level three in a raster scan pattern between pre-set human way points to generate a complete map of the environment. In an unstructured and unknown environment a human generated path may not be discernible or optimum for reliable mapping, but in this case ground truth will be known and way points can be easily set. This experiment will serve as a comparison to the amount of navigation information a common navigation technique stores compared to the learned CAM-SLAM information stored as a CAM.

3.5.3 Experiment three: Learning a change in the environment

In order to test CAM-SLAM's ability to adapt to a change in the previously static environment it has already learned, the doors to CO346 and CO350 (see figure 3.6) were opened. The open doors presents the robot with a new potential action to take and changes the associated LIDAR data the robot has already learned in the *initial learning experiment*. Using the previously learned $[P]$, 10 further learning iterations are run to test how CAM-SLAM will adapt to this change.

3.5.4 Experiment four: Sharing navigation information through CAM-SLAM

Two experiments are designed to test the ability of CAM-SLAM to share navigation information between heterogeneous robotic platforms, Ownbot and the Pioneer. In these experiments each of the robots will be placed

in the opposite environment and be given the CAM learned by the other robot.

The first experiment will test whether navigation information learned by a lesser sensory capable robot (Ownbot) can be useful for a relatively more sensory capable robot (Pioneer) and *vice versa*. This experiment will run both of the robots with LIDAR sensors, with each LIDAR having different sensing capabilities and positions on their respective platforms. This means the robots can match sensor data to match decision points during navigation, potentially providing a more robust classifier matching criterion.

The second experiment will test the performance of CAM-SLAM sharing navigation information learned from two robotic platforms utilising different sensor types and SLAM techniques. The novel part of this experiment is sharing information between a LIDAR based platform using GMapping and a camera based platform using RatSLAM. Each respective SLAM technique is designed to operate with each sensor type, and sharing information between them would currently require human interaction. This is because it is not possible to directly compare 2D distance data from the LIDAR, with 2D colour data from the camera. Furthermore, GMapping generates a metric grid based map of the environment, and RatSLAM generates a Cartesian point map of the path the robot takes within the environment. This requires a means of converting between each perspective, which is a non-trivial problem. CAM-SLAM seeks to solve this problem through learning a map of decision points (CAM) using actions as a common perspective between the robots.

Chapter 4

Cognitive Action Mapping SLAM Results

The previous chapter detailed the CAM-SLAM algorithm, which is designed to learn a map of decision points within an environment. It is hypothesised that by learning a map of decision points CAM-SLAM can: reduce the amount of unnecessary mapping information, reduce the memory costs of maps, and allow navigation information to be shared between heterogeneous robotic platforms. This chapter presents the results of training CAM-SLAM on real-world robotic platforms and evaluates the performance of CAM-SLAM based on these criteria.

The experiments in this chapter have trained the system on two heterogeneous robotic platforms (Ownbot and Pioneer in section 3.4.1) on levels two and three of the Cotton building (see section 3.4.2). This is to provide a comparison on the performance of CAM-SLAM learning on robots with different capabilities, and on robots utilising different SLAM approaches. Each experiment has been run through 10 *trials* for repeatability testing and statistical analysis using the Student's T test. Ideally, a minimum of 30 trials would be required for statistical analysis, but these experiments are being run on real-world robots and 30 trials requires an impractical amount of hours to complete. This is due to the time constraints of run-

ning autonomous robots in an active work space, the limited battery life of the robots, and requiring personnel to be present to observe the robot in case of failures or potentially hazardous malfunctions. However, the Student's T test is designed for small sample sizes. A small sample size will not differentiate the trials if there are small differences but will indicate large differences between trials.

Each *trial* is a run of 10 *learning iterations*. A *learning iteration* is a single traversal of the robot from the start location to the goal location, one iteration of the ACS process. This is based on previous work which has shown localisation improvements occurring within 10 *learning iterations* in similar work [103]. Between each *learning iteration* only the CAM (classifier populations) is kept, the full SLAM map (mental map) from each respective SLAM approach is discarded at the end of each iteration. It is expected that only the learned navigation information in the CAM is required for effective navigation.

To judge the performance of CAM-SLAM, the CAM for each experiment is compared to a human determined ground truth of the environment. Ground truth is determined by judging regions within the environment where the robot will be required to make a decision about the action it will follow. The ground truth for the levels two and three of the Cotton building were presented in figures 3.5 on page 68 and 3.6 on page 69.

4.1 Learning a CAM with Ownbot

The aim of this experiment was to evaluate CAM-SLAM's ability to generate useful CAMs. This was done individually on two heterogeneous robotic platforms to test the robustness of the system to varying degrees in measurement uncertainty, in both the sensors and movement of the robots. The second part of the experiment was to evaluate CAM-SLAM's ability to learn with both a range and vision based mental map.

To achieve this, CAM-SLAM was trained on the Ownbot and Pioneer

platforms on levels three and two of the Cotton building respectively. First, both robots were trained with their respective LIDARs and utilised GMapping for the mental mapping stage. Second, the Pioneer was trained again, on level two, this time using its camera and RatSLAM for the mental mapping stage. Note, both Pioneer experiments were started from the beginning with no information shared between experiments.


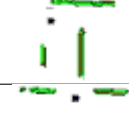


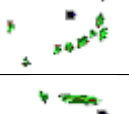
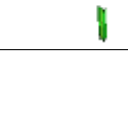
4.1.1 Experiment one: Ownbot

Considering the ground truth map for level three of the Cotton building (figure 3.6 on page 69) it can be determined that there are five decision states Ownbot must identify (out of an infinite possible in the continuous real environment) and take action to reach the goal location following an optimal path. The optimal path is the path that requires the robot to take the least number of actions to reach the goal location autonomously.

From 10 trials the average number of classifiers CAM-SLAM learned to successfully path-plan to the goal location through the shortest path is six. This population is shown as an LCS population in table 4.1. Each classifier has a condition containing locally relevant information about the decision point and an associated expectation link indicating the expected classifier (decision point) should the robot take a given action. This is shown in table 4.1 in the *Expectation* column, where the action has the expected classifier ID underneath, e.g. given classifier one if you take action north you would expect to reach classifier two. Finally, the classifier contains a fitness value indicating its usefulness for navigation. A high level of fitness (i.e. approaching one on a zero to one scale) demonstrates that these classifiers have a relatively high prediction accuracy. The higher the prediction accuracy the more likely it is to be correct and therefore the more the robot can trust it for path-planning.

From the final populations, the top five classifiers have an average fitness rating of 0.82 with a variance of 0.15. The high level of fitness across each trial indicates that CAM-SLAM is repeatedly learning and re-

Table 4.1: Classifier population learned to lead to the goal location on the third floor of Cotton.

Classifier ID	Condition	Expectation (Action → Classifier ID)				Fitness
		North	East	South	West	
1		2	N/A	N/A	N/A	0.90
2		7	3	1	N/A	0.88
3		8	5	9	2	0.77
4		8	5	9	2	0.82
5		10	N/A	6	3	0.79
6		5	N/A	N/A	N/A	0.75

associating meaningful decision points within the environment that are useful for navigation.

The population can be displayed as a CAM by placing the classifiers relative to the position estimates in each classifier's condition. This is shown in figure 4.1 on page 78, where the black squares indicate position, green lines indicate LIDAR data, and the numbers indicate the classifier's ID (labelled one through six). These six classifiers had the highest fitness level across the final population after 10 learning iterations. When compared to the map generated by GMapping in figure 4.2 on page 79, the CAM is mostly empty of features or details. Only the key decision points

within the environment have been mapped.

Comparing to ground truth, Ownbot generated one extra classifier, competing classifiers three and four, than the considered ideal of five. This is due to a data association problem, where inaccuracies in the localisation of the robot's angular position (given the relative low accuracy of its odometry) lead to poor matching performance between classifier states. Therefore, poor matching in this particular region consistently generated two competing (but still useful for navigation) classifiers. The reason this particular area is prone to odometer errors is when coming towards the region from a westerly (relative to the robot's actions) direction there is a short slope in part of the corridor. Because of this viewing the location from each direction the robot's sensors will be skewed relative to each other. This causes some inconsistencies in the mental mapping approach which adds error to the position of the robot.

As would be expected, the first classifier in each trial had a fitness rating of almost one. This was because the robot's (and the CAM-SLAM's) starting position was always pre-set by a human and sensor data could match the static environment. CAM-SLAM was always able to match the first state and did not produce any separate competing classifiers. CAM-SLAM on average across the 10 trials, produced three separate classifiers per each of the five human judged decision states. States further from the goal location, in terms of distance, produce on average one more classifier than classifiers at closer states. This was due to closer classifiers having a higher confidence in their position estimate giving better classifier associations and *vice versa* for more distant classifiers.

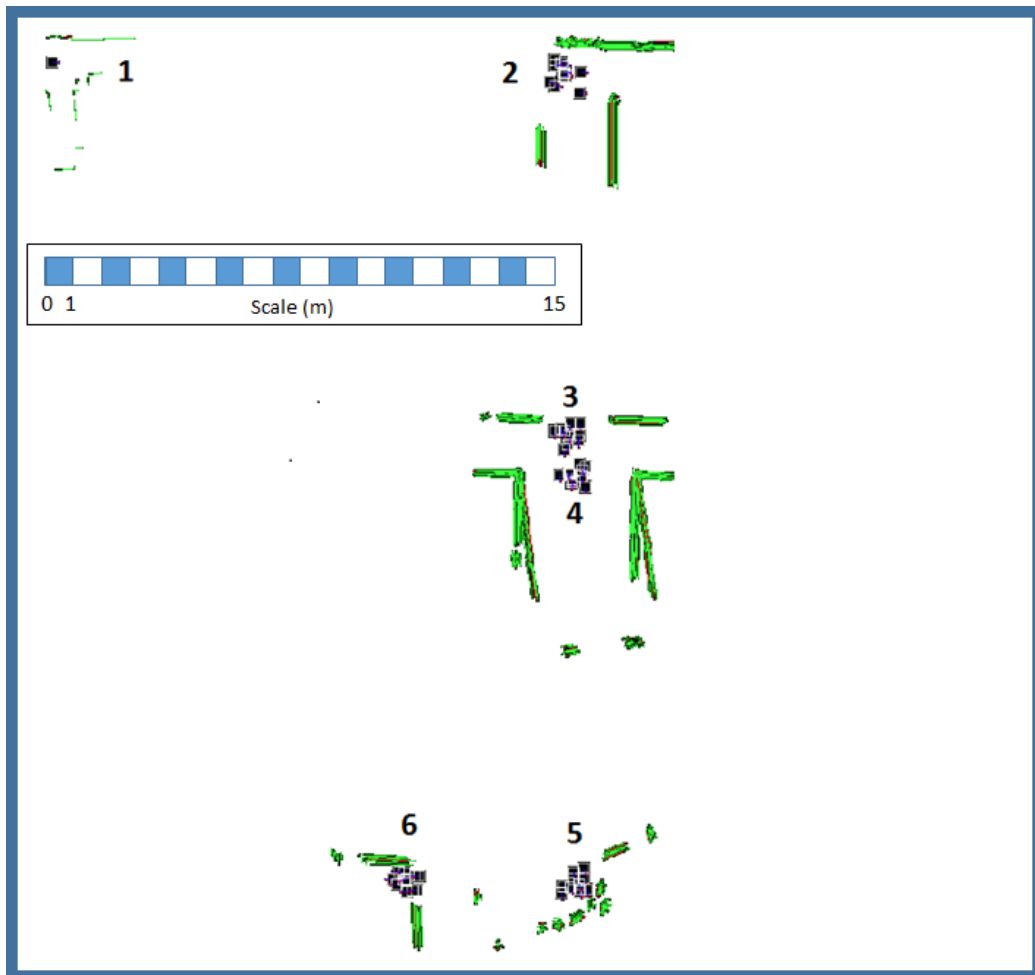


Figure 4.1: Top six classifiers, for each of the 10 trials, which were learned to lead Ownbot to the goal location on the third floor of the Cotton building. Note that only decision points are stored as opposed to the complete mapping information.

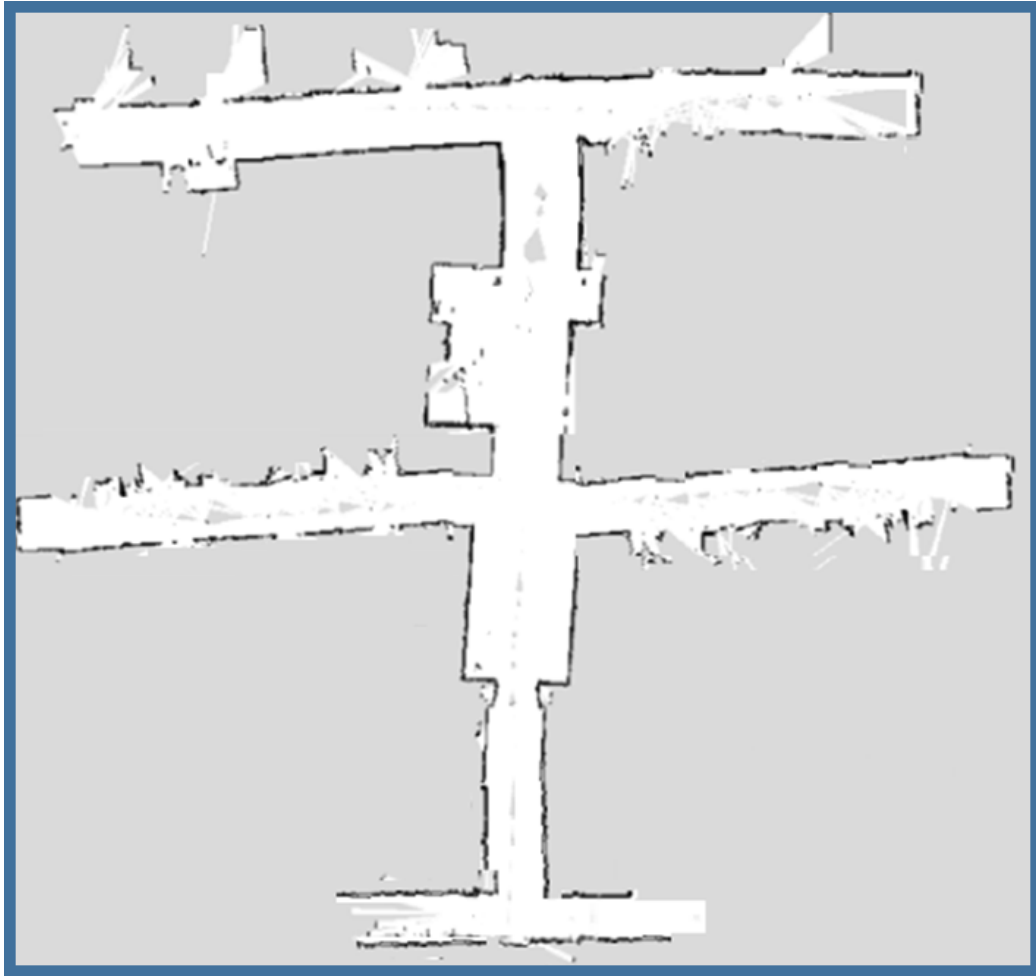


Figure 4.2: Mental map generated by GMapping during the final learning iteration of CAM-SLAM with Ownbot on the third floor of the Cotton building, where the corridors are actually orthogonal in the world.

4.2 Learning a CAM with Pioneer

4.2.1 Experiment one: Pioneer

The Pioneer was subject to two separate experiments on level two of the Cotton building. First, it was trained with its LIDAR using GMapping. The first experiment provides a comparison of CAM-SLAM when trained on a relatively more sensory capable robot when compared with Ownbot. Second, it was trained with its camera using RatSLAM. The second experiment is to evaluate CAM-SLAM operating with a vision based mental mapping stage (SLAM). No information was shared between each experiment, the Pioneer learned from the beginning of both experiments.

4.2.2 Pioneer: GMapping mental mapping stage

Considering the ground truth map for Cotton level two (figure 3.5 on page 68) it can be determined that there are five decision states the Pioneer must identify and pass through to reach the goal location following an optimal path. From 10 trials the average number of classifiers CAM-SLAM learned to successfully path plan to the goal location through the shortest path was five. The decision points consistently matched the human judged ground truth. From the final populations, the top five classifiers had an average fitness rating of 0.88 with a variance of 0.11. The high level of fitness across each trial indicates that CAM-SLAM is repeatedly learning and re-associating meaningful decision points within the environment. When compared against Ownbot, the classifier matching performance was higher. This is due to the Pioneer's relatively more accurate odometry, and more accurate LIDAR allowing CAM-SLAM to match decision points more robustly.

The CAM produced by the Pioneer is shown in figure 4.3, with the map generated by GMapping shown in figure 4.4 on page 82. Similar to Ownbot, the CAM has significantly reduced the amount of information

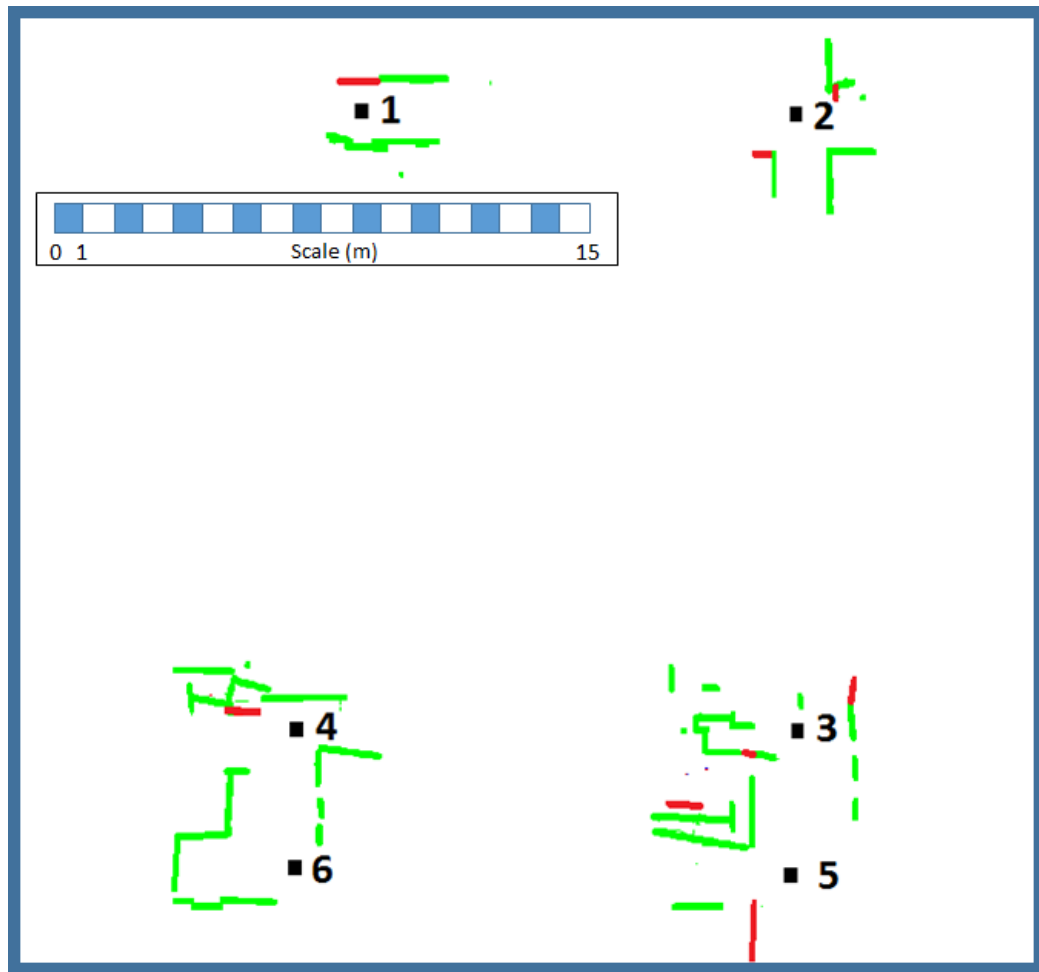


Figure 4.3: LIDAR based CAM learned to lead Pioneer to the goal location on the second floor of the Cotton building.

required to navigate to the goal location.

4.2.3 Pioneer: RatSLAM mental mapping stage

In this experiment the Pioneer utilised RatSLAM for localisation, and generated decision points in CAM with image data. This means that CAM-SLAM was matching decision states based on image comparisons instead of range data. After 10 trials on level two of the Cotton building the aver-

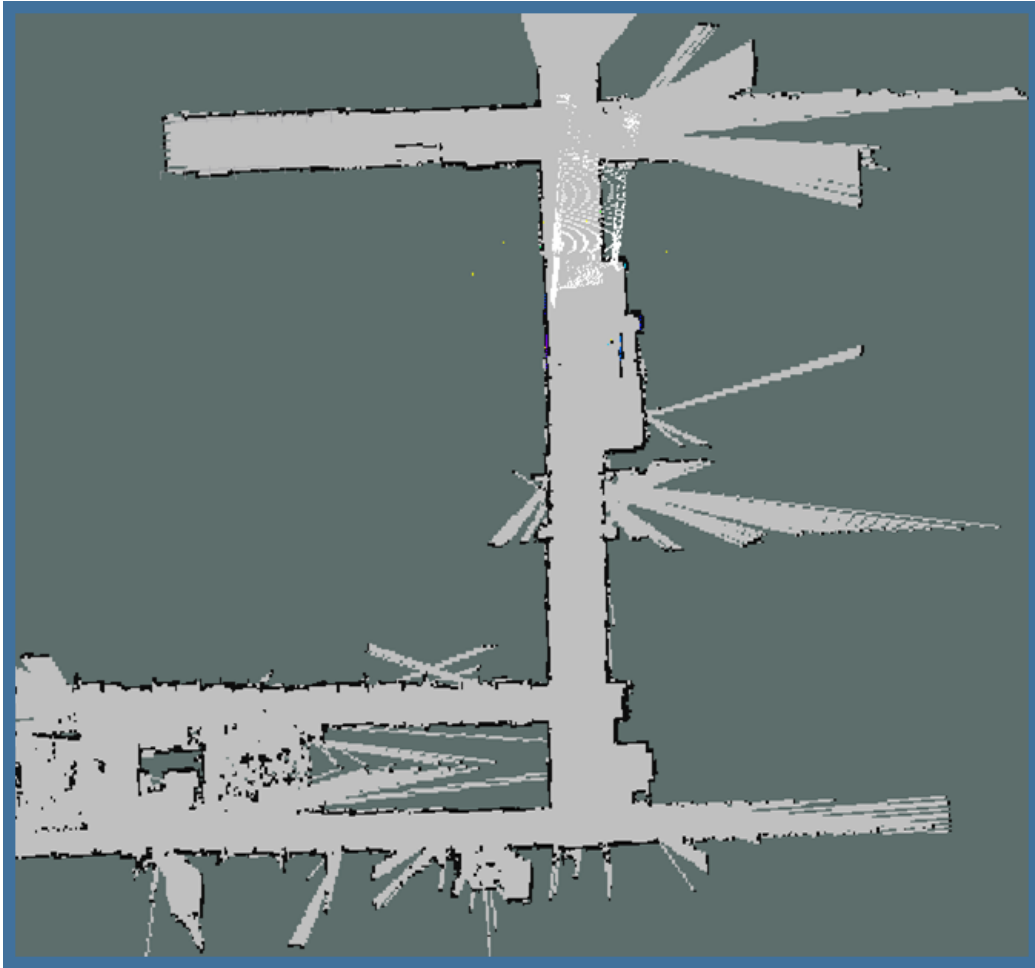


Figure 4.4: Mental map generated by GMapping during the final learning iteration of CAM-SLAM with the Pioneer on the second floor of the Cotton building.

age number of classifiers CAM-SLAM learned to successfully path plan to the goal location through the shortest path was five. The decision points consistently matched the human judged ground truth.

The CAM produced by the camera can be seen in figure 4.5. Each classifier is represented by up to four images, one per available action. An enlarged example of this is shown in figure 4.6 on page 85, where the classifier is at an intersection with four possible actions. Similar to the LIDAR based approach, these classifiers are the only navigation information created about the environment, with no detailed image maps being created. As before, an infinite number of images could be used for navigation. The robot must recognise a decision point based on the learned images, actions, and position estimates only.

From the final populations the top five classifiers have an average fitness rating of 0.84 with a variance of 0.19. The high level of fitness demonstrates that CAM-SLAM is capable of repeatedly learning meaningful image based decision points within the environment that are useful for navigation.

4.2.4 Summary of initial learning

The initial learning experiments have demonstrated the ability for CAM-SLAM to learn a CAM that is capable of directing a robot to a given goal location. CAM-SLAM has shown good performance at autonomously generating classifier populations (CAMs) that can be used for path planning and mapping of a real-world environment on a physical robot, seen by the high fitness of the classifiers. CAM-SLAM learned classifiers which were capable of predicting which state to expect next, if a robot were to take a given action, and provide a fitness rating for the robot to gauge which path is the most reliable from its own experience. This was successful when both a LIDAR and camera based SLAM approach was used as the mental mapping stage. The advantage this provides is each robot can utilise the best suited SLAM approach for its given capabilities with

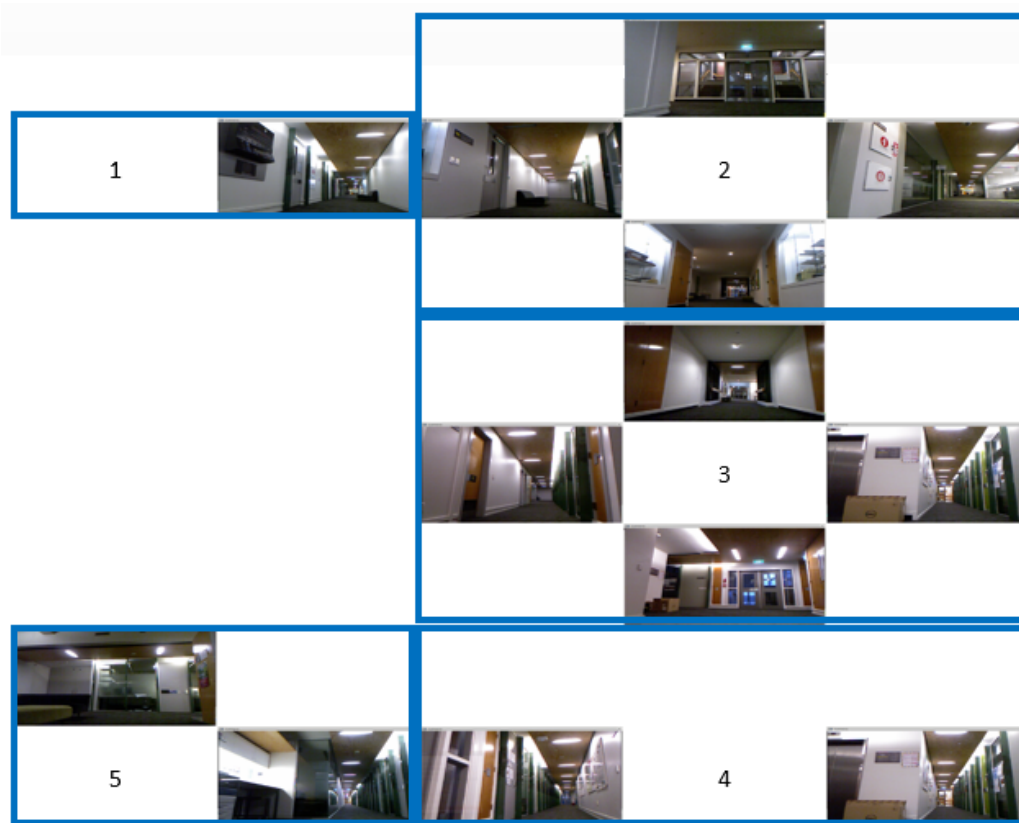


Figure 4.5: Image based CAM learned to lead Pioneer to the goal location on the second floor of the Cotton building.

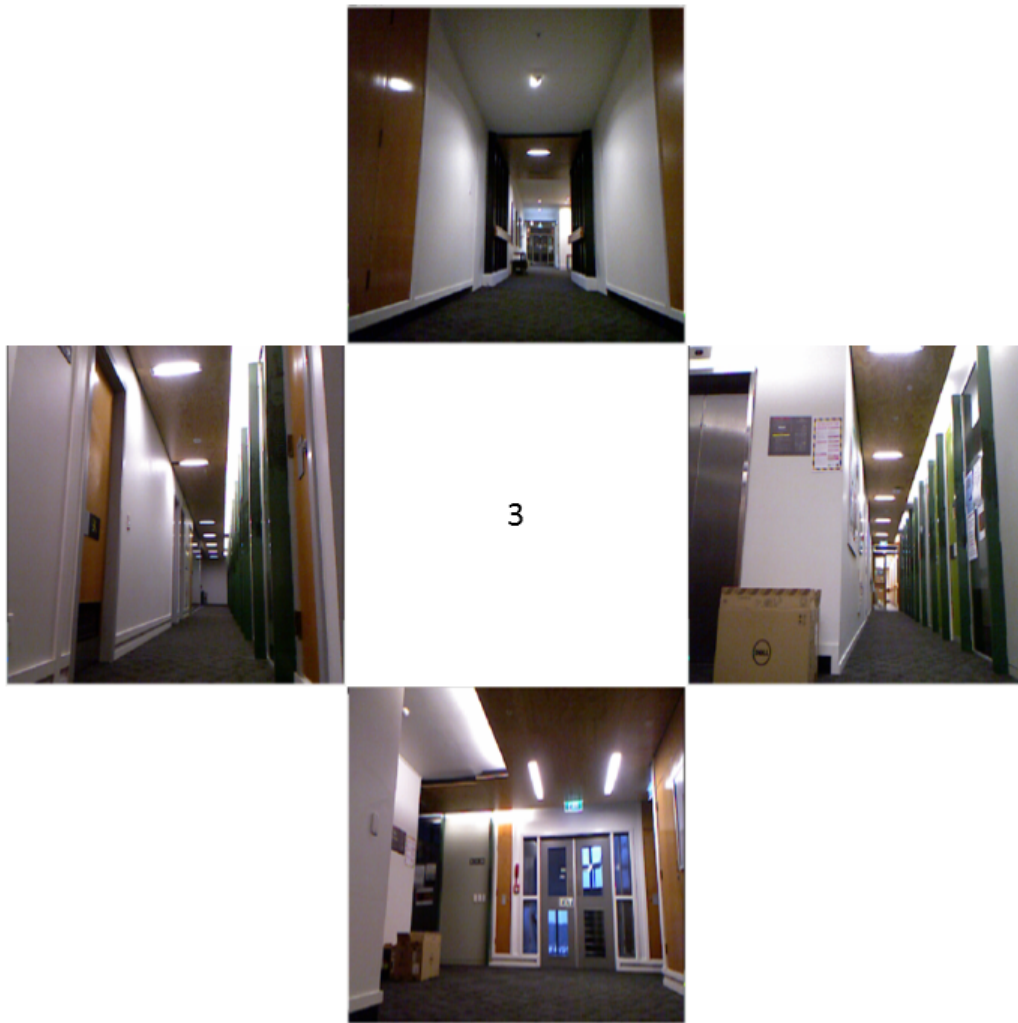


Figure 4.6: Enlarged example of a image based decision point learn by the Pioneer on the second floor of the Cotton building.

CAM-SLAM. The next experiment will evaluate if the CAM is storing less navigation information than a standard SLAM approach.

4.3 Benchmark Algorithm experiment

In order to quantify the reduction in navigation information; the memory size of a complete map of an environment generated by GMapping was compared with the CAM generated by Ownbot. Ownbot was manually driven by remote control through Cotton level three along the centre of the corridors to generate a complete map of the environment. In an unstructured and unknown environment a human generated path may not be discernible or optimum for reliable mapping. In this case ground truth was known and way points could be easily set.

The GMapping (driven by a human) generated map can be seen in figure 4.7, with an overall memory size of 45 MB. Conversely, in CAM-SLAM each population across 10 trials had an average memory size of 1.5 MB with a variance of 0.3 MB. CAM-SLAM was capable of repeatedly navigating a learned environment without any human knowledge, and reduced the overall memory requirements to successfully navigate by 97%. Although this experiment was run on a relatively small environment size compared to modern SLAM approaches, the reduction in memory usage will scale to larger environments provided the technique scales.

4.4 Experiment three: Learning a change in the environment

To test the performance of CAM-SLAM when a change in the environment occurs, Ownbot was required to navigate Cotton level three with its CAM (from section 4.1.1) again but with CO350 (shown in figure 3.6 on page 69) now open. This change affects the LIDAR associations and provides the

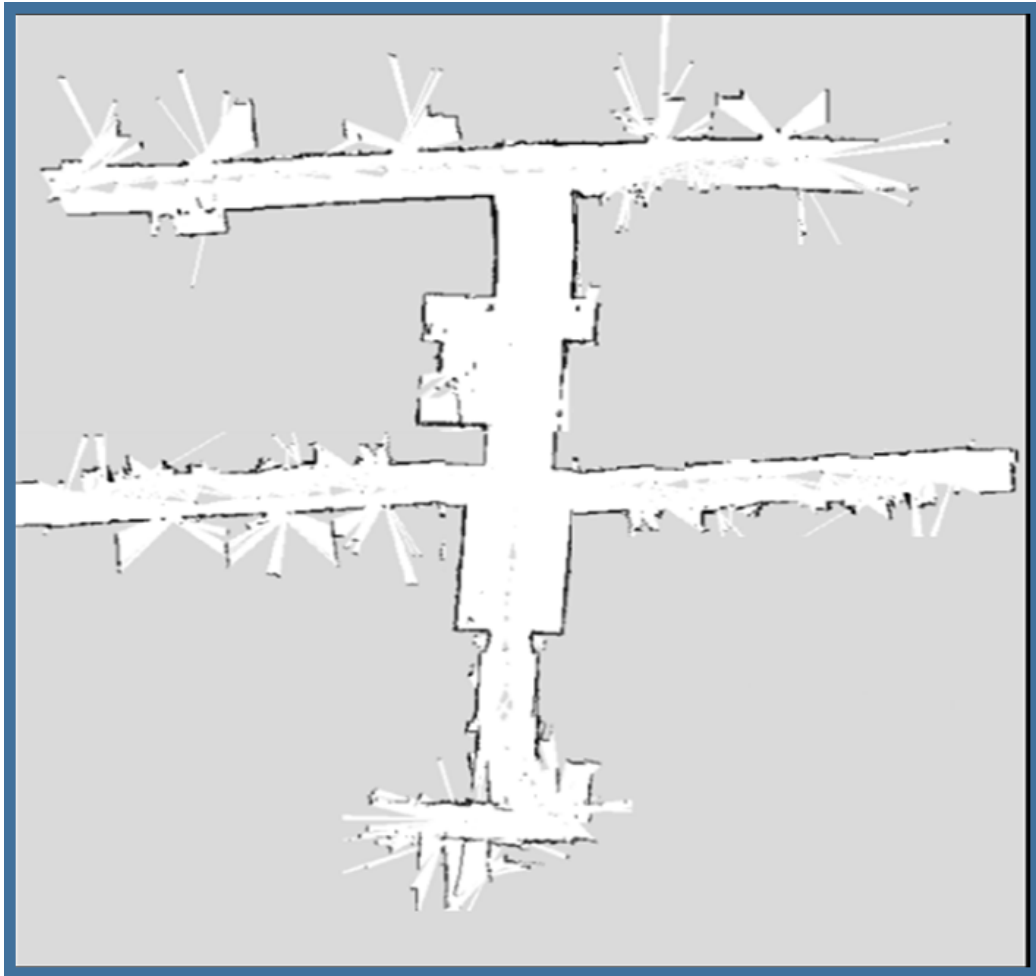


Figure 4.7: Human generated map of the third floor of the Cotton building.

robot with two new actions to learn.

The change on the third floor of the Cotton building affected the previously learned CAM in three ways, see figure 4.8. First the initial classifier cl_1 , figure 4.1 on page 78, was no longer correctly matching the new sensor condition. However, the position estimate still matched, thus a new classifier cl_8 was formed by the system, initialised at half the fitness rating of cl_1 . As training progressed, cl_8 became the dominant classifier as the previous classifier cl_1 could not match the new sensor condition. cl_8 was then able to learn to expect cl_7 when taking action *east* and cl_2 when taking action *north*, see figure 3.6 on page 69.

However, cl_7 , was still able to associate the sensor condition as the open door did not drop the matching rate below the set threshold of 95%. cl_7 did learn to expect cl_8 when taking action *west*, however its sensor condition cl_s still indicated an obstacle. This does not affect CAM-SLAM's path planning between classifiers but does create an inconsistency when viewed in the map. The main reason for this is CAM-SLAM does not have an 'infeasible' expectation flag, i.e. if an action is not feasible the classifier does not learn to expect either itself or an 'infeasible' state. This allows CAM-SLAM to learn expectations when an action becomes feasible.

The final effect was cl_2 having to re-learn the expectation from taking action *south*. cl_2 originally learned to expect cl_1 however cl_8 became the dominant classifier. After 10 learning iterations cl_2 still predicated cl_1 , this showed that it did not have sufficient training to re-learn the expectation to the new classifier cl_8 .

CAM-SLAM did struggle to re-learn after a change in the environment, but it was still functionally capable. Classifier matching conditions may need to be stricter on sensor matching conditions to allow CAM-SLAM to recognise changes in previously learned sensor data. However, an overly strict matching condition will increase the number of classifiers generated due to sensor noise and minor variations in the real-world, increasing the required learning time. Data association is a common problem in LI-

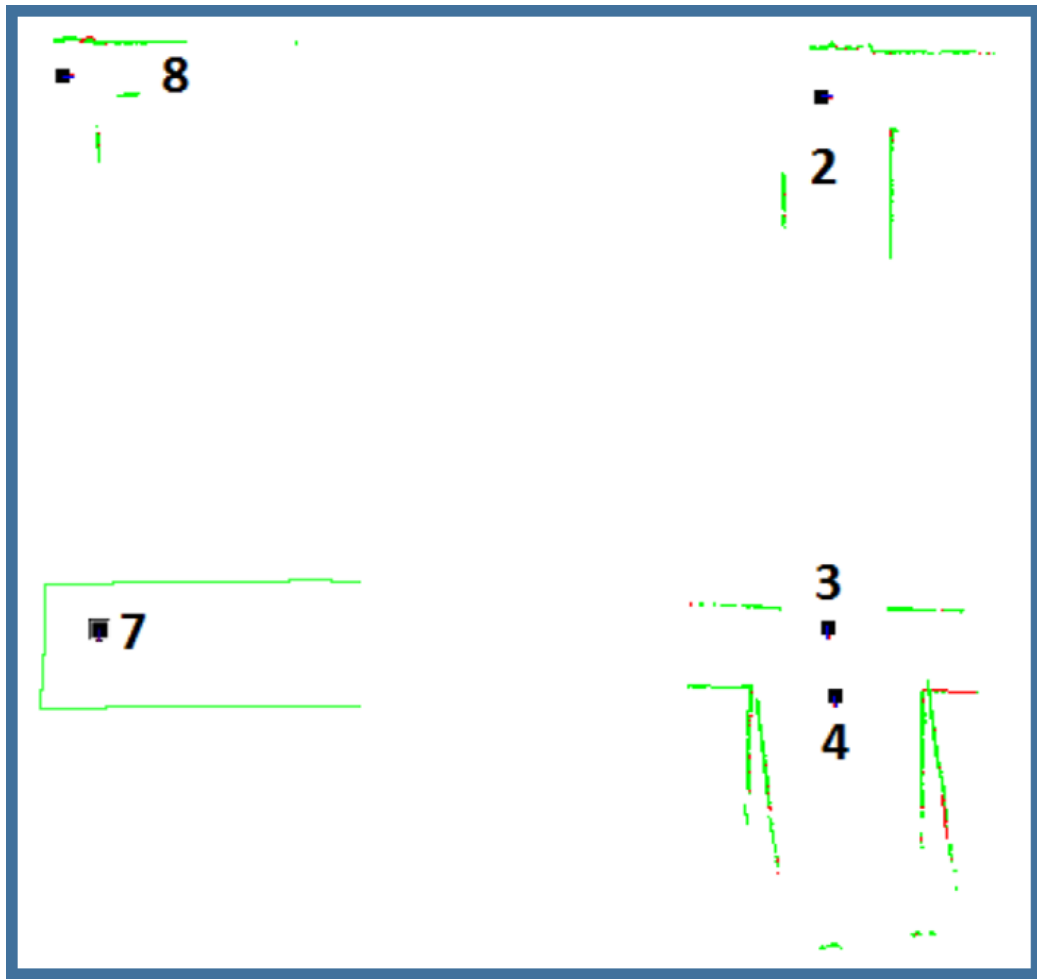


Figure 4.8: Subsection of the CAM learned after a change to the static environment on the third floor of the Cotton building.

DAR based navigation [39] and CAM-SLAM does not provide a means to improve LIDAR associations. However, CAM-SLAM was only required to partially re-learn two decision points from the Pioneer in order for it to navigate to the goal location. This demonstrates that CAM-SLAM is robust to these problems as it still navigated to the goal and shared appropriate knowledge. The next set of experiments seek to evaluate the performance of CAM-SLAM sharing CAM between two heterogeneous robotic platforms.

4.5 Experiment four: Sharing navigation information through CAM-SLAM

The experiments conducted now seek to demonstrate CAM-SLAM sharing navigation information between heterogeneous robotic platforms through CAMs. The first experiment demonstrates a CAM being shared by Ownbot to the Pioneer, where each has relied on their respective LIDAR sensors. That is, a CAM generated by a robot with a low capability LIDAR (low cost) is being shared with a robot with a high capability LIDAR (high cost). This ability to share information from a low cost robot to a relatively high cost robot can be useful in hazardous co-operative robotics domains, such as disaster relief. The low cost robot can be used for initial investigation and mapping before risking the higher cost robot.

The second experiment shows the CAM sharing navigation information between the GMapping approach on Ownbot and the RatSLAM approach on the Pioneer. That is a range-based to camera-based robot, where both the sensors and morphologies are different. A direct one-to-one mapping at the sensor level is therefore impossible. A higher-order abstraction in terms of the CAM produced in the previous results is necessary.

4.5.1 Same sensor type, different morphology experiment

The Pioneer was tasked with navigating level three of the Cotton building using its LIDAR and GMapping as the mental mapping stage. The Pioneer was given the CAM generated by Ownbot in section 4.1.1, i.e. CAM-SLAM's classifier population was seeded with the classifiers learned by Ownbot. However, given the different sensing capabilities and position of the LIDARs for each platform, matching the Pioneer's state to the classifiers generated by Ownbot is not guaranteed.

Considering Ownbot's CAM (figure 4.1 on page 78), the Pioneer had an initial population of classifiers (one through six) from Ownbot, with which to plan a direct path to the goal location. The best case scenario is if all the classifiers in the seeding population will be matched by the correct Pioneer state and no new classifiers would be required from the Pioneer. This will require CAM-SLAM to correctly match the state of the Pioneer to the condition of the classifier generated by Ownbot. The consequence of creating new classifiers is that ACS has to learn new expectations, which increases the learning time.

From the 10 trials on the Pioneer, the top classifiers now in the CAM are shown in figure 4.9. The CAM learned by the Pioneer does not contain the redundant classifier from the initial CAM and now matches ground truth. Through better quality LIDAR and odometry estimates, the Pioneer was able to repeatedly match classifier three at that location in the environment. As a result, the expectation mapping of neighbouring classifiers was able to learn to expect classifier three. This is a positive result as it demonstrates the ability for CAM-SLAM to continue learning across heterogeneous robots.

Across all 10 trials the Pioneer was able to reliably match classifiers one, two, and three. On average each classifier was matched in eight learning iterations with a variance of one, creating on average four new classifiers with a variance of two. However, in each of the 10 trials the Pioneer struggled to repeatedly match classifiers five and six. On average classifiers five

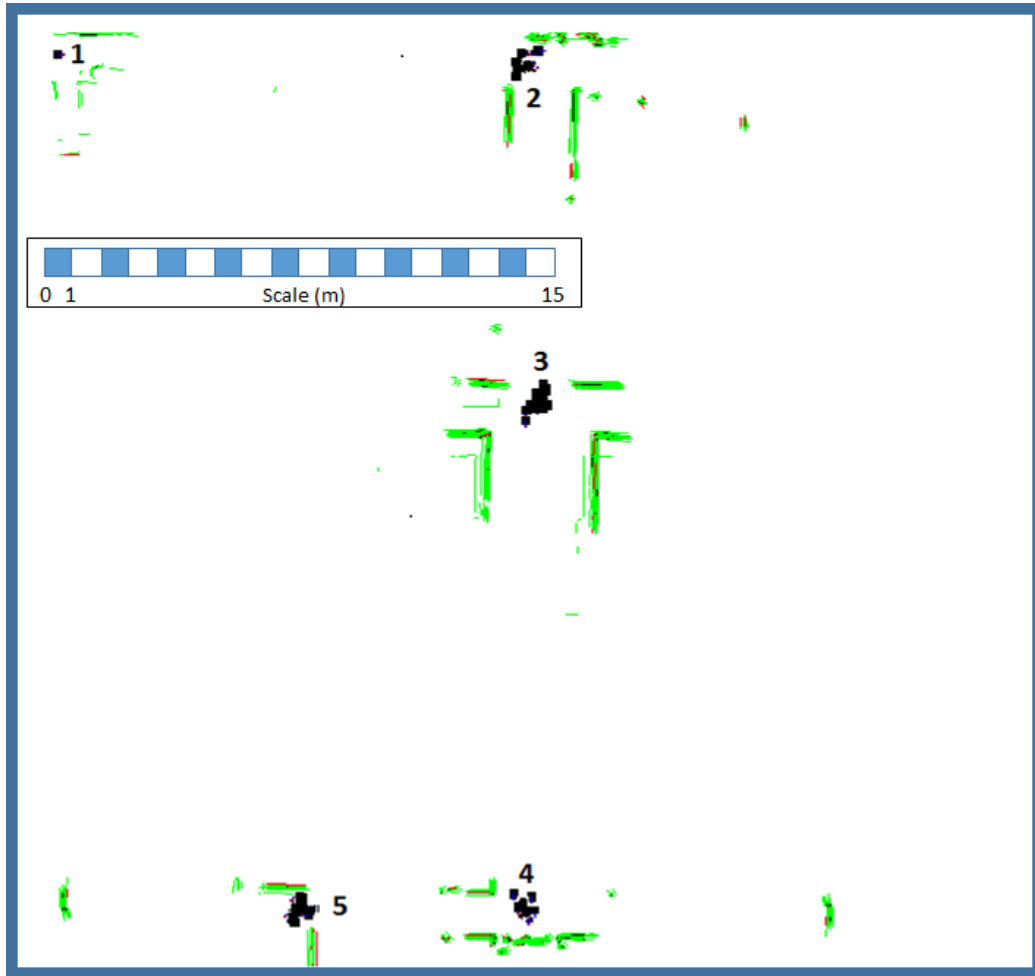


Figure 4.9: LIDAR based CAM learned by sharing information between Pioneer and Ownbot on the third floor of the Cotton building

Table 4.2: Average number of matches for Ownbot Classifiers across 10 trials

Classifier ID	1	2	3	4	5	6
Matched	9 \pm 1	7 \pm 0.5	8 \pm 0.7	3 \pm 1.2	4 \pm 0.9	2 \pm 2

and six were only matched four times with a variance of two, creating on average seven new classifiers with a variance of two. Table 4.2 shows the average number of times the Pioneer platform was able to match each classifier learned by Ownbot shown in figure 4.1 on page 78. Table 4.3 shows the average number of classifiers created for each of Ownbot’s classifiers.

LIDAR association does not rely on the co-ordinate of the robot’s position; however it does rely on the angular position estimate for data comparison. Ownbot’s position estimate was more accurate at the start of the navigation compared to later into the navigation for each learning iteration. This produces classifiers with an accurate angular estimate in the early stages of the learning and *vice versa* for later classifiers. An analysis of the classifiers shows that classifiers one through three have a high confidence in position estimate, with an average variance of 0.2 m. This provides better matching performance for classifiers on both LIDAR data and position estimates. Classifiers five and six had a low confidence in their position estimate with an average variance of 0.6 m. This reduces the matching performance of the LIDAR data through an unreliable angular estimate and makes position matching unreliable.

These results suggest that a CAM-SLAM is strongly dependant on its mental mapping stage, requiring accurate angular estimates to generate similar sensory perspectives. With CAM-SLAM’s current sensor matching methods, if the size of the domain is increased then robots with poor localisation performance will not be able to share navigation information successfully. If CAM-SLAM were to be applied to larger domains, a pose-invariant feature extraction solution is required. Computationally, CAM-

Table 4.3: Average number of new classifiers created by the Pioneer for each of Ownbot's classifiers across 10 trials

Classifier ID	1	2	3	4	5	6
New Classifiers	2 \pm 1.2	6 \pm 2.0	4 \pm 1.2	2 \pm 0.6	7 \pm 2.1	7 \pm 1.8

SLAM scales linearly with the number of classifiers added and can scale provided the mental mapping approach can provide a localisation estimate.

Classifier four, a redundant classifier created by Ownbot, was only matched an average of three times with a variance of two across the 10 trials. A qualitative analysis of the ground truth suggests that classifier three was a better match for the LIDAR data from the Pioneer than classifier four. The consequence of this is CAM-SLAM was able to learn to expect classifier three, in place of three and four and hence reduce the number of classifiers, seen in figure 4.9 on page 92, needed to path plan to five which now matches the ground truth.

The time required for Ownbot to learn the third floor of the Cotton building over a trial was on average one and half hours with a variance of 10 minutes. The time required for the Pioneer to do a trial with the prior learning from Ownbot was on average half an hour with a variance of 10 minutes. The Pioneer learning on its own, with no prior learning, in the same environment had an average trial time of an hour with a variance of 15 minutes. Contrasting with the Pioneer's own individual learning run, Ownbot's learned classifiers improved the learning time of the Pioneer platform and did not adversely effect its learning, even with the conflicting classifiers.

This result demonstrates that sharing navigation information between the robots is beneficial, as the time required to learn was shown to improve. This is beneficial as multiple lesser capable (cheap preferably dis-

posable) robots can learn multiple environments in parallel at low cost or risk. The information can then be shared with the more capable (expensive) robot in each of the environments, reducing time spent in the environment, reducing risk to the robot and improving productivity.

4.5.2 Different sensor, different morphology experiments

These experiments will test the ability for CAM-SLAM to share navigation information between a LIDAR and a vision based robot system. Two experiments will be run, the first will run the same robot with different sensors to isolate the sensor as the main difference between the robots. Using the same robot means the movement uncertainties from the robot's odometry estimates are the same for each SLAM process. This means that the only difference in positional estimate should be from the inherent differences between RatSLAM and GMapping estimates. The second experiment will share a CAM between two robots with different sensors and morphologies. This will require CAM-SLAM to match states with both different sensor types, and positional movement uncertainties.

The first experiment tasked the Pioneer to navigate Cotton level two with its camera and RatSLAM, provided to the Pioneer was the CAM generated with its LIDAR and GMapping in section 4.2.2. Given the different sensors CAM-SLAM had to rely on matching decision points based on action matching (see section 3.3) and position estimates (see section 3.3). After 10 trials the resulting CAM is shown in figure 4.10. The result is five decision points, matching ground truth, that have both the previously learned LIDAR data and new image data. This result shows that CAM-SLAM has successfully collated the information from both RatSLAM and GMapping into a single CAM, by learning decision points with both LIDAR and image based conditions.

In the second experiment the CAM-SLAM had to rely on action matching and position estimates for classifier association, but this time from relatively less reliable estimates on the position from Ownbot. Using its

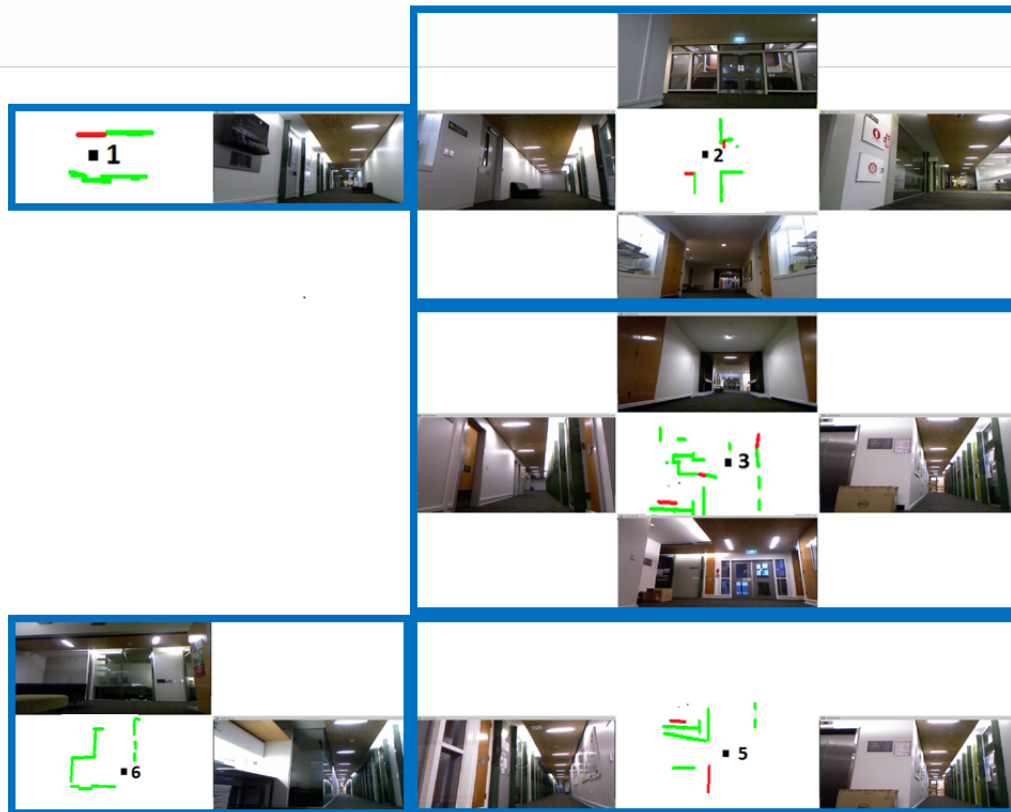


Figure 4.10: LIDAR and image based CAM learned by the Pioneer on the second floor of the Cotton building. The classifiers contain both the LIDAR data and image data for each decision point that was learned by the Pioneer.

camera the Pioneer was tasked with navigating Cotton level three seeded with the CAM learned by Ownbot in 4.1.1. After 10 trials the resulting CAM is shown in figure 4.11. The result is five decision points, matching ground truth, which are a mix between classifiers containing both image and LIDAR data, and the original LIDAR based classifiers.

Classifier three is a new classifier learned by the Pioneer with the camera, where initially the seeding CAM had two competing classifiers. An analysis of the learned populations shows CAM-SLAM only partially matching either of the competing matching classifiers, leading the new classifier to become the dominant classifier through consistent sensor matching.

Conversely, classifier five is an original classifier from the initial CAM. From the classifier population the fitness of the original classifier is consistently higher than any of the newly formed classifiers for that decision point. It appears this is due to low image matching performance for that particular region of the environment, due to varying lighting conditions by a nearby window. With poor image matching the robot has instead relied on the action and position matching for classifier association, which has further improved the fitness of the original classifier. This is a benefit of the CAM-SLAM approach, whereby sensor matching is not required for navigation if the robot is sure of its position. This can account for dynamic environments where a particular region can change frequently, e.g. common work space. However, relying on position estimates in the current implementation could hinder learning times as new classifiers are being generated to compensate for the potential change in the environment. Image matching performance can be improved through better place recognition algorithms, which will help in dynamic environments to reduce the reliance on position matching.

One of the limiting factors of this approach to sharing information is the re-learning of the expectation mapping. The expectations are still required to be learned for classifiers neighbouring new and potentially useful classifiers. This was seen as relatively newer classifiers were consis-

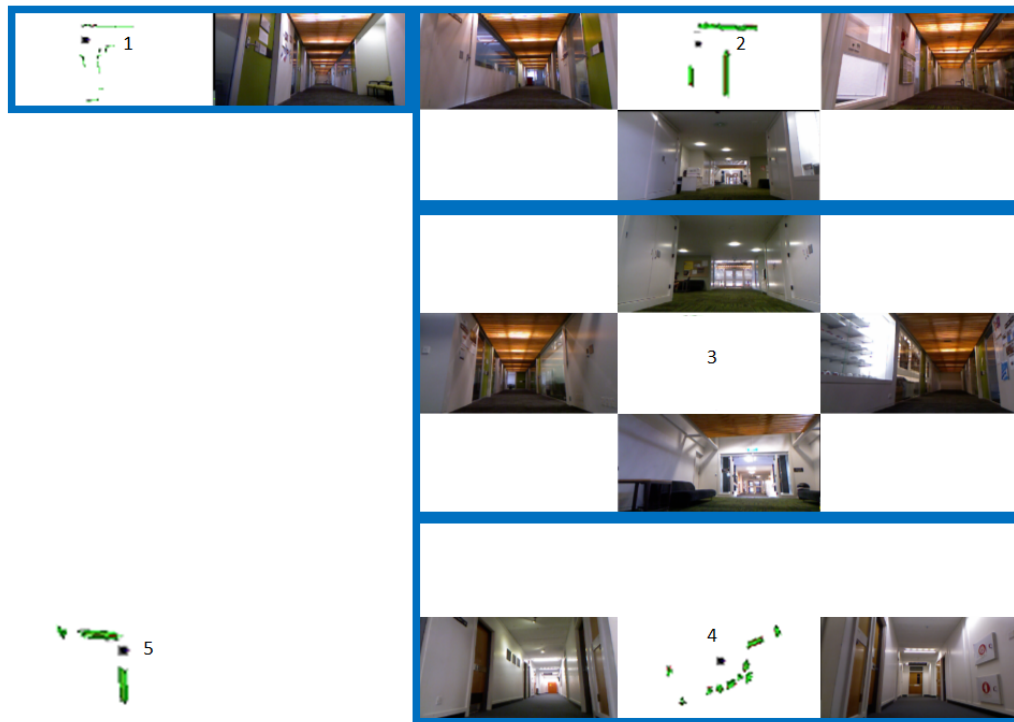


Figure 4.11: LIDAR and image based CAM learned by the Pioneer on Cotton level three. Classifier three contains only image based data, but has condensed the original six classifiers from Ownbot into one. Classifier five only contains LIDAR data, as image comparison in that region was not reliable but classifier fives action matching and position estimate were.

tently being matched (indicating their usefulness for path planning), but not yet strongly being expected by neighbouring classifiers, limiting the performance of any future path planning. A potential solution to this is propagating new classifiers, generated from position only matching, to neighbouring classifiers to allow classifiers to seed expectations.

4.5.3 Summary of shared learning

CAM-SLAM has shown to be capable of sharing navigation information between two robotic platforms with different sensors and morphologies through CAMs. This was demonstrated by the repeatability of the classifier matching and the reduction of classifiers learned by the Pioneer. The sharing of information has also shown; improved learning times for robots with seeded CAMs, continued learning through combining sensory information from two robots, and the capability to share less reliable sensory data with more capable robots.

4.6 Summary

CAM-SLAM, through ACS and generalisation, was shown to autonomously generate CAMs capable of effectively and consistently navigating robots to given goal locations with 97% less mapping information than standard approaches.

The results have demonstrated the successful sharing of CAMs between two heterogeneous robotic platforms in a real-world environment without excessive relearning, compared with learning a CAM from scratch. Furthermore, this result has demonstrated the novel contribution of combining navigation information from two distinct SLAM approaches (GMapping, and RatSLAM) into a single CAM useful for both robots with LIDAR or vision based sensors. Sharing of navigation information between two robotic platforms is a positive result as it allows robots to share and make

use of their respective learning capabilities within environments.

Chapter 5

Emotion inspired adaptive robotic path planning

The previous two chapters have discussed CAM-SLAM, a topological map based system for sharing navigation information between heterogeneous robotics platforms. CAM-SLAM provides high-level path planning information through decision points but does not supply a means for robots to navigate between decision points at a local level. To navigate between points a robot will require a navigation system that can determine a path through the environment taking in locally relevant information.

Ideally a robotic navigation system should adapt its path planning and behaviour to overcome a variety of obstacles within an environment without the need for specialised stimulus-response planning approaches. These specialised approaches can not be capable of handling every possible situation, with the number of required stimulus-response patterns being potentially infinite. Fellous [33] suggests that biological emotions facilitate the transferring of simplified, but high-impact information, both externally (e.g. facial expressions) and internally, for operating system-like tasks (e.g. reacting to 'threatening' situations). This concept of emotions is appealing for path-planning as it can potentially be used to generalise the web of stimulus-response links in computational path-planning tech-

niques. This is beneficial as it provides an adaptive path-planning approach without the need to account for every possible scenario the robot will encounter.

Inspired by these theories on natural emotions (see section 2.4), this chapter presents a novel emotion model designed to beneficially adapt a rigid computational path-planning approach (see navigation stack section 2.8). Rather than mapping external stimuli directly to responses, the emotion model will learn an intermediary set of emotion categories. This method is intended to achieve many of the same path-planning goals but in a more generalised way, i.e. the external stimuli to response cases are filtered through the emotions to provide a compact set of rules or policies for the robot to follow.

In engineering and biology, the concept of a bow-tie is used to represent complex adaptive systems in a way that provides flexibility without compromising efficiency. The shape of a bow-tie describes systems that include large numbers of inputs and outputs bridged by a smaller number of intermediary states and processes. Emotions may form the knots of some of these bow-tie structures in biological cognitive systems, decoupling stimulus and response [65]. An important function of emotions in this model is to provide a compact set of high-impact behaviour adaptations, reducing an otherwise tangled web of stimulus-response patterns into a compact and manageable structure. The hypothesis here is that the emotions compact the search space and reduce the complexity (total number of state-action pairs) of the system. Where the environmental search space is the map of all possible external stimuli to responses.

The objective is to develop a method to learn these emotional responses as the robot interacts with the environment. Humans have a bias on what they define an emotion to be, e.g. sadness, and how it affects (beneficially or not) the behaviour of an agent (e.g. a robot). Not all emotion responses are intuitively considered beneficial, but removing them from the system could potentially be limiting the behaviour of the robot. For ex-

ample, sadness or grief appears to prevent a person or animal from being mobile or productive [49], in robotics this response would be considered a hindrance. However, sadness is believed to provide a means of focusing cognitive learning processes onto a specific problem, by shutting out any further stimuli from the environment [114]. Through learning such responses, rather than pre-specifying them, this work seeks to prevent human biasing and instead allow emergent responses to form. However, learned emotions will not necessarily match their human counterparts, as the problem of mobile robot navigation differs from many problems that human emotions have evolved to address. The learned bow-ties can still provide an interesting insight into the formation of emotions.

5.1 Emotion model

An emotional response is designed as a bow-tie structure linking emotional reinforcers (stimuli, see section 5.1.1) to behavioural modifiers (response, see section 5.1.2) through an intermediary internal emotional state. A visual representation of this bow-tie structure is given in figure 5.1. The system may have multiple bow-ties that function in parallel but only one bow-tie (emotion) may be active at a time. This is consistent with cognitive science research, where it is commonly told that only one natural emotion can be active at one time [87].

The goal of the emotion model is to learn useful behavioural responses, through these bow-ties, that aid successful navigation to given locations without prior input or bias (such as human determined responses to the emotion states). In order to learn these bow-tie structures an Accuracy Based LCS (XCS) will be used as a transparent global search technique with classifiers in the form of Reinforcer-Emotion-Modifiers.

Crucially, the emotion states in the bow-tie are not pre-set with a defined emotion (e.g. happy or sad) or modifier (response). This is in contrast to known prior work, such as [58], that pre-sets the stimuli to defined

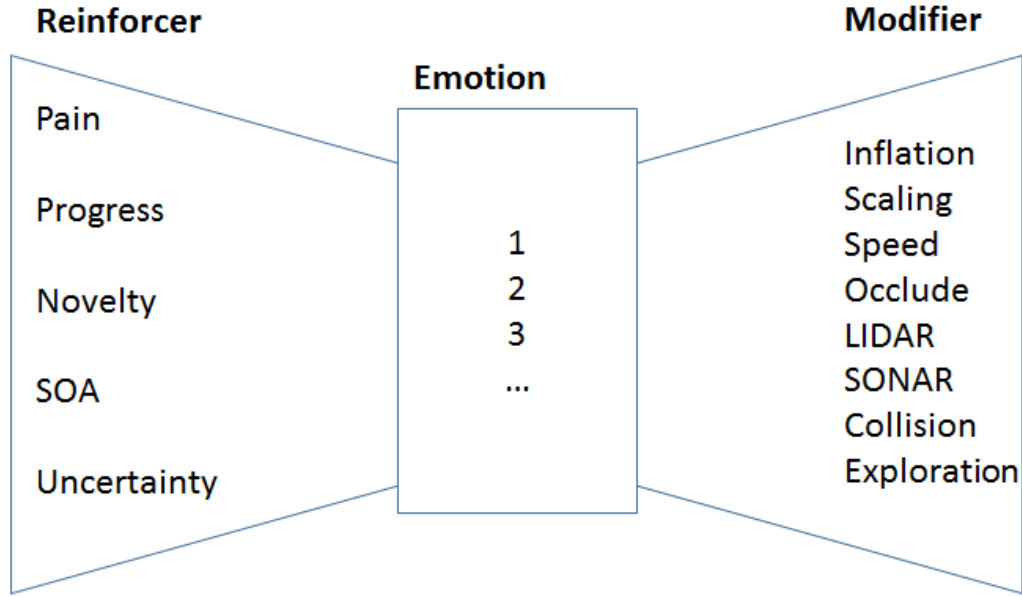


Figure 5.1: Reinforcer-Emotion-Modifier bow-tie structure

emotion states and then defined behaviours considered associated to that emotion. Instead, this work will attempt to learn these behaviours as the robot navigates the environment. This means the system will be learning to match emotions to modifiers as the modifiers are being learned. The system is co-evolving the modifiers and the reinforcer to modifier links, this is shown in figure 5.2.

Based on the modifier each emotion selects, the resulting navigation behaviour can be used to label the emotion in a *post hoc* anthropomorphic manner. Although interesting, the labelling has no functional contribution, with any performance benefits still present if the emotion remains unlabelled (does not have a human equivalent emotion). As an example, ‘pain’ as a reinforcer can associate an emotional response to an aspect of an environment, such as an obstacle. The resulting action taken by the robot is considered to be the behavioural response for the emotion. The behaviour associated to the example emotion may be to avoid the obstacle to mitigate pain from collisions. We could then label this emotion ‘fear’

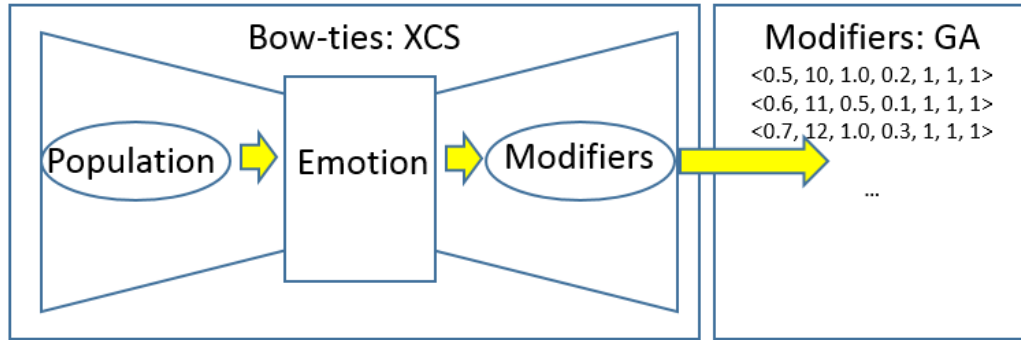


Figure 5.2: The bow-ties are learned via the XCS algorithm, while the modifiers are learned by the GA. Both learning algorithms receive reward when the robot reaches a goal location

retrospectively, based on the learned (evolved) behaviour after the fact.

5.1.1 Reinforcers

Methods are required to convert sensor readings to perceptions, fed as reinforcers to the learning system. Rolls [86] [87] presents a dimensional model in which emotions result from the presence, omission or termination of rewards and punishers, which he calls instrumental reinforcers. Instrumental reinforcers can be primary, determined by genes (e.g. the tastes and smells associated with food), or secondary, learned by association with primary reinforcers (e.g. the sight of a favourite food). From the large, but non-exhaustive list of primary reinforcers suggested by Rolls [86], members of a smaller subset were selected that are likely to be beneficial for robotic navigation:

- Robot *pain* could be chosen to represent potential damage resulting from collisions. Biological pain can occur without any other observable signs of tissue damage [98]. Similarly, robot pain should not require actual damage, only a collision. Thus pain is induced via bump sensors at the front and rear of the robot, and is calculated as -1.00 for no collision and 1.00 for a collision with no modifier for

severity.

- Novelty is the quality of a phenomenon being new, original, or unusual. Novelty may be the shared experience of a new phenomenon or the subjective perception of an individual. This reinforcer encourages an animal or robot to seek out and investigate unexpected or poorly-known objects or areas. We calculate this as the fraction of new regions n seen by the robot's sensors compared to previously seen regions s .

$$novelty = n/s \quad (5.1)$$

The map is represented as a grid, which is updated with sensor information at each time step. The number of unknown grid cells being updated (seen) at each time step gives n . The total number of grid cells within the sensor range gives s . Entering a new region will cause an increase in novelty as more regions are mapped.

- Progress is the robot's internal estimate of its progress towards completing the current goal. This is calculated via the local path planner as the ratio of the current number of completed way points over the total number of way points planned to reach the global goal (goal location).
- Sense of Agency (SOA) is a measure of how effectively an animal or robot is able to control external events through its own actions [18]. Rolls [86] lists "control over actions" as a primary reinforcer, but this could also be linked to the 'active versus passive response' dimension used to determine whether to elicit active emotions such as fear or anger, or more passive ones such as sadness [86] [87]. SOA is represented as a success to failure ratio of all goals delivered to the controller, e.g. whether the robot reaches the goal successfully or

must abort its current objective. The number of successes and goals sent are tracked by the system and SOA is calculated as:

$$SOA = success/goals \quad (5.2)$$

the higher the success rate, the higher the SOA.

- Uncertainty is defined as a lack of information about an event, or how uncertain an animal or robot is about its current course of action. For robotic navigation, uncertainty is calculated as the estimate of localisation accuracy from the SLAM process within the robot's navigation system.

Each reinforcer tracks the current value (proportional) at each time step, the integrated value (integral) over all time, and the rate of change (differential) of the reinforcer. To achieve this, each reinforcer's value is scaled between -1.00 and 1.00 . The integrated term represents the influence of the reinforcer over time on the emotional state, building over time to potentially elicit a response. For example, if pain is building over time (e.g. from multiple collisions) it would be potentially useful to elicit a response to slow down or increase avoidance parameters. The rate of change of the reinforcer represents the current trajectory of the reinforcer, depending on its upward or downward trajectory different responses may be elicited. For example, if pain is reducing over time it may not be advantageous to immediately respond to a potential collision. These examples are not programmed into the system, but are mere considerations of what could be a useful response from an emotion system. The system is to autonomously learn the associating modifier for each emotion state.

5.1.2 Modifiers

A modifier is the term that denotes a set of parameters that will adjust the robot's navigation behaviour within its environment. The modifier parameters influence the paths generated by the robot's navigation algorithm in an attempt to improve the robot's progress towards the given goal location.

The navigation system we use is from Robot Operating Systems (ROS) navigation stack described in section 2.8. The navigation stack is a commonly used robotic navigation algorithm that provides path planning and navigation controls for generic robotic platforms, with a variety of parameters for influencing navigation. Typically these parameters require tuning for individual robotic platforms. A modifier is a set of these navigation parameter values that influence the generated navigation behaviour from the navigation stack.

The navigation stack utilises a grid cell map of the environment for path planning, created using the robot's external sensors and localisation through GMapping (see section 2.8). The grid cell is a map of costs associated to regions within the environment, where obstacles seen by the robot will add cost to the grid cell. Path-planning approaches then attempt to find a path of least cost through the grid cells to a goal location, i.e. the path of least resistance. An example of the grid based path planning is shown in figure 5.3. The darker regions in the grid have a higher path-planning cost, while lighter regions have a relatively low cost.

The modifier parameters influence the paths generated by the navigation stack by altering the formulae that generate the cost of the grid cells. This causes regions to change their respective costs of obstacles, freeing or restricting the robot's navigation. The set of modifier values used to influence the navigation stack are:

- The *inflation radius* is the maximum distance from obstacles at which a cost to path plan should be incurred. For example, setting the in-

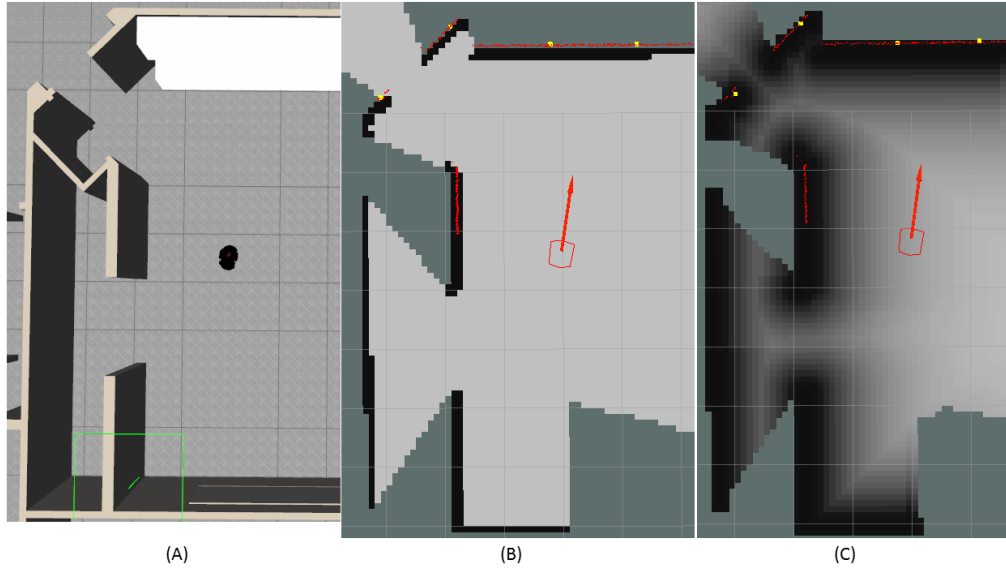


Figure 5.3: Navigation Stack mapping example. (A) Willow Garage simulation (B) Low cost path planning (C) High cost path planning.

flation radius at 0.55 m means that the robot will treat all paths that stay 0.55 m or more away from obstacles as having equal obstacle cost. All areas between the obstacle and the inflation radius will be given a score relative to how close to the obstacle the path is, with higher costs for paths that lead closer to the obstacle. The range can be set to any value, however it is limited to between 0 m and 30 m for this work.

- The *cost scaling factor* determines the cost of the paths between an obstacle and the inflation radius. The lower the cost scaling factor, the more the cost is spread, see figure 5.3 as an example. With a low cost scaling factor the cost is spread further away from the obstacle, resulting in low cost paths being pushed further away from obstacles. This scaling factor is limited between 0 and 30. This is calculated as,

$$e^{-1.0 * \text{cost_scaling_factor} * (\text{distance_from_obstacle} - \text{inscribed_radius})} * (251 - 1) \quad (5.3)$$

where the inscribed radius the size of the robot platform.

- *Speed* is the maximum speed the robot can move in metres per second, with the Pioneer's maximum speed reaching 4.00 ms^{-1} . The minimum speed of a moving robot is set at the default setting of 0.10 ms^{-1} , to prevent behaviours which can not move.
- *Occlude distance* is the weighting for how much the controller should attempt to avoid obstacles when path planning. *Occlude distance* increases the cost score of planned paths that traverse through any high cost regions. This primarily affects paths which may be low cost (generally shorter paths), but require some traversing of regions close to obstacles. The occlude distance is limited between 0 and 1. The occlude distance is an addition to the cost calculation for each individual path:

$$Path_{cost} = Map_{cost} + Occlude * (maximum_obstacle_cost) \quad (5.4)$$

- To generate the grid map used to navigate, each sensor (LIDAR, SONAR, Collision) is used to update the map based on its sense of the obstacles in the environment. Each sensor is allocated a weighting, between 0 and 1, to determine the strength of the obstacle within the map. If a sensor is allocated a high weighting, the path planning will be more cautious about the obstacles found by that particular sensor.

An example of a modifier is shown in table 5.1, using the navigation stack's default parameter values. Each modifier also has an associated fitness score to determine how useful it is for the robot's navigation performance. The fitness score is updated based on the reward provided by the environment for reaching the goal location, with beneficial modifiers having a higher fitness score.

Table 5.1: Modifier Example, showing default values from the Navigation stack.

Inflation radius	Cost scaling	Speed	Occlude distance	LIDAR	Sonar	Collision	Fitness
0.55 m	10.00	1.00 ms^{-1}	0.02 m	1.00	1.00	1.00	1.00

In summary, this section has described each part of the Reinforcer-Emotion-Modifier bow-tie model. The next section will describe how these bow-ties can form analogies of human emotion.

5.2 Anthropomorphic emotions

In Plutchik's [83] model, basic emotions have a subset of *specific emotions*. These *specific emotions* are caused by different reinforcer triggers but elicit the same *basic emotion*. For example a person can be 'happy', but more specifically they could be 'satisfied' or 'comfortable'. Table 5.2 outlines a simplified set, suitable for proof of concept, of reinforcers and the emotions that they may trigger.

Typical reinforcement triggers described by Rolls are presentation, omission, and termination; but this excludes a fourth possibility representing a transition from omission to presentation, which may be just as relevant as termination for some reinforcers/emotions. As an example, the detection of an altruism cheat might elicit a disgust/dislike response. However, if the cheat was a formerly trusted individual, an anger/betrayal response is more likely. The important characteristic in the model is the state change (the rate of change), which could be in either direction.

Provided is a list of emotions that could potentially emerge from the emotion model and how they could benefit the robot's behaviour:

Happiness is considered to often be triggered by goal achievement, the receipt of a reward, or the absence of an expected punishment. Happiness encourages an animal or human to repeat rewarding (or punishment-avoiding) behaviours. Positive emotions associated with happiness

Table 5.2: Emotions and reinforcers that could potentially trigger them.

Emotion	Specific Emotion	Instrumental Reinforcer			Expected Resposnes
		Presented	Omitted	Reversal	
Happiness	Satisfaction	Progress			Increased Speed
	Relief		Pain		Decreased obstacle costs
Sadness	Depression	Pain	Progress		Decreased Speed
		Uncertainty	SOA		Increased obstacle costs
Anger	Frustration	SOA	Progress	Yes	Decreased obstacle costs
Fear	Caution	Pain			Decreased Speed
		Novelty			Increased obstacle costs
Suprise	Shock	Uncertainty		Yes	Decreased Speed Increased obstacle costs

have also been linked to a broadening of cognitive focus [46], directing attention outward and encouraging interaction with the environment [8]. In a robot, this could be represented as increased mobility (higher speed) and reduced cost applied to the strategies that led to success.

Sadness is often associated with the termination of a highly-valued reward (e.g. loss or grief), the failure to achieve goals or the receipt of punishments when only a passive response is possible [86]. Sadness facilitates introspection, slowing responses and directing attention inward, potentially gaining insights that might improve future performance. It also has an important social role, in that expressions of sorrow encourage others to provide assistance to the distressed individual [8]. An appropriate response for a mobile robot might be to reduce its speed, reorganise or reset its internal maps and/or request assistance.

Anger is typically elicited when an attempt to reach a goal is obstructed [78] and an active response is possible [87]. Anger increases activity and decreases risk-aversion, allowing the obstruction to be overcome by force. In a robot, this can be expressed as decreased safety mar-

gins, reducing obstacle avoidance and/or increasing speeds.

Fear often occurs when a punishment is expected. It increases risk-aversion and encourages the avoidance of, or escape from, dangerous stimuli. In humans and animals, the avoidance response to fear is likely to be all-consuming, suppressing high-level reasoning to a greater extent than distress/sadness [97]. Most robot architectures lack the cognitive flexibility to benefit from such a heavy-handed approach. A typical mobile robot controller, such as the subsumption architecture [10], is planning and reacting simultaneously, and both layers already have all the computational resources they need at all times (otherwise the robot would crash). Diverting resources from one layer to the other would likely be counterproductive. A more appropriate response for a mobile robot is simply to increase its safety margins and/or move away from a fear-inducing object.

Surprise is triggered when there is a mismatch between perception and prediction, for example discovering a novel region in a familiar area. Surprise interrupts ongoing activity [97], directing attention toward novel stimuli. The shock or startle response normally associated with surprise could cause a mobile robot to decrease speed or increase safety margins.

The system is not designed to learn these emotions explicitly, however it has the capability to produce these forms of responses. Empirical analysis of the bow-tie structure will be used to judge if any of the learned emotion states are analogous to a human emotion. The next section will provide details on the implementation of the emotion model in the XCS.

5.3 Implementation

A novel Reinforcer-Emotion-Modifier classifier population is learned through an XCS to represent the emotional model's bow-tie structure shown in

figure 5.1 on page 104. The Reinforcer-Emotion connection is similar to the standard State-Action classifier representations, in that the reinforcer (state) elicits an emotional response (action). The Emotion-Modifier connection is a secondary State-Action classifier, where the emotion (state) elicits a behavioural response through the modifier (action). The emotion state is acting as a secondary state to compact the initial reinforcers into a manageable discrete set of emotions for learning a behavioural response.

Figure 5.4 shows this classifier structure compared to a standard XCS and ACS classifier representation, see section 2.5.3. The figure shows a robot in an example state in an environment $\langle 1, 1, 1, 0, 0, 0, 1, 1 \rangle$, with the one representing a wall and a zero representing an open space. The standard XCS classifier representation is a State-Action rule which will take an action in where its condition matches the current state, in the example the robot will take action four. In ACS the classifier will take an extra step and predict the effect of taking the selected action. In the example the classifier predicts the next state will be $\langle 1, 1, 1, 0, 0, 0, 0, 0 \rangle$.

This implementation is a co-operative evolution between the modifier population and the classifier population. The modifier population is evolving useful navigation behaviours as the robot navigates. The classifier population is learning to match reinforcer states to a modifier (behaviour) through an intermediary emotion state. The aim is to generate emotional responses beneficial for robotic navigation. Both populations are learned as the robot navigates to given goal locations.

During navigation the robot's emotional reinforcers are stimulated through its interaction with the environment. From the reinforcing input an emotion state is evoked from the model. Note, at all stages the robot will have an emotion state, even if it is currently considered a low quality response. The current emotion state then elicits a modifier, from a set of modifiers, which affects the behaviour of the robot (navigation stack path-planning). This process is shown in figure 5.5 on page 116.

The XCS algorithm has been adapted to handle the Reinforcer-Emotion-

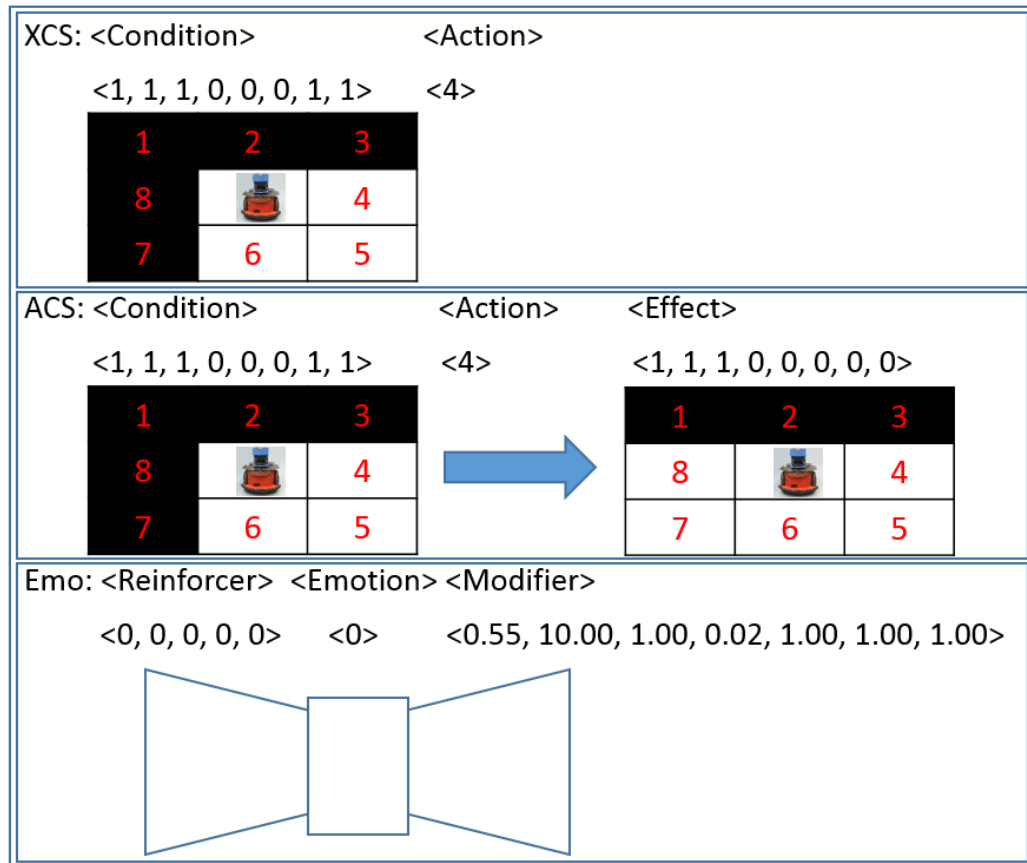


Figure 5.4: Standard structure of an XCS and ACS classifier compared to the emotion based classifier. In emotion (emo) based classifier the emotion is a secondary condition, used to compact the initial reinforcer condition, and the modifier acts as an action.

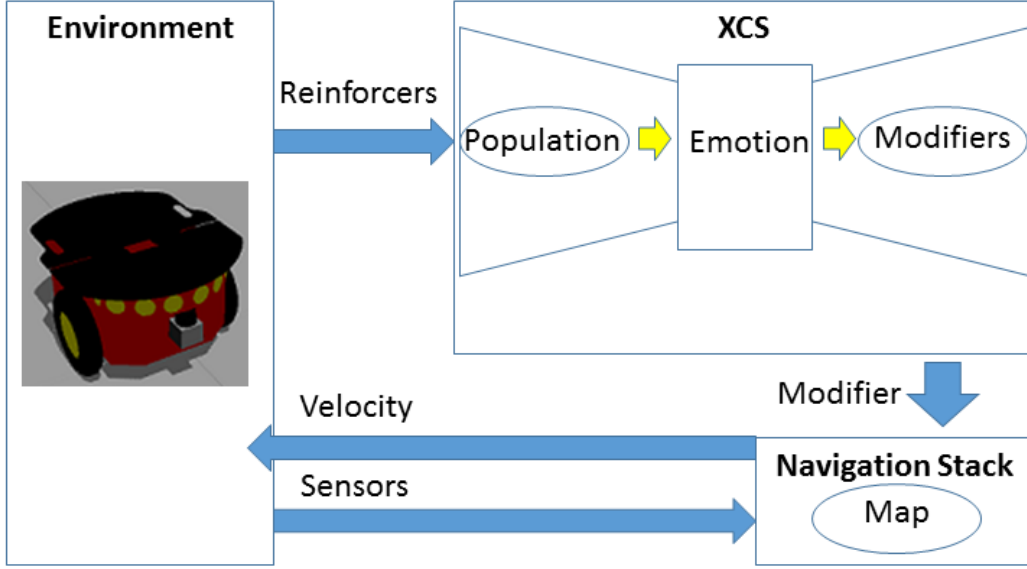


Figure 5.5: The robot's interaction with the environment stimulates emotional reinforcers, which elicits an emotion state in the robot. The emotion state then elicits a modifier affecting the navigation stack, adjusting the behaviour of the robot's navigation.

Modifier classifier and follows a similar process as described in section 2.6. XCS is typically used to learn complete mappings of problem environments, through learning classifiers that are accurate in their reward prediction. However, this work is exploiting XCS's capability to generalise by learning bow-ties in the form of compact human readable classifiers. The XCS will attempt to learn a complete mapping of the emotion model, however what that would be or mean in terms of emotion states is unknown so it is not a major goal of this work. Instead an in-complete emotion model that is still beneficial for navigation will be considered a success.

5.3.1 Reinforcer Input

In each iteration the robot's reinforcers are stimulated by the environment generating a new reinforcer state R . If R is different from the previous iteration's state R_{t-1} it is sent to the XCS to generate the match set $[M]$, else the system will continue with the previously chosen action. $[M]$ is formed

by comparing R with each classifier in the population. As each reinforcer contains real world values, a classifier's condition contains an upper and lower bound around each value. The upper and lower bounds of the value are pre-set to ± 0.20 of the values in R during the covering stage. This value was empirically set to allow a range of matches to be possible without the system being too strict. Being too strict causes the system to change emotion states frequently, as the matching stage becomes overly precise. For a classifier to be considered a match, the current reinforcer state values must be between the upper and lower bounds of the respective values in the classifier's condition.

$$lower \leq value \leq upper \quad (5.5)$$

The 'don't care' (#) value is the equivalent of a maximum upper and minimum lower bound for each value of the condition, such that any value will match. If $|M|$ is below the minimum size threshold the *covering stage* is called. As per standard XCS, the covering stage creates a new classifier using the current R to generate the condition and associates a randomly selected emotion.

5.3.2 Reinforcer-Emotion

From $[M]$ a classifier is selected to elicit its associated emotion e . Typically this is done by alternating between the classifier with the highest prediction value, and a random selection to balance exploration and exploitation. However, by inspection it was found that this led to connections with early rewards dominating the population. In order to allow exploration and to prevent over-fitting the first classifiers, for the first 100 successful goal locations the selected classifier is always chosen at random from $[M]$. After this, the selected classifier alternates between the Reinforcer-Emotion connection (most fit classifier) and roulette wheel selection.

5.3.3 Emotion-Modifier

The selected e is then used to select a modifier mod in the modifier set $[Mod]$. If $|Mod|$ is below a maximum size threshold a new modifier is generated. Modifiers are initially created at random, with each parameter being set to a random valid value. Once the $[Mod]$ is full, every 20 iterations a GA is used to generate new and potentially useful modifiers in the population (as is common practise for $[P]$ in XCS).

The emotion states have a connection value con_{em} associated to each individual modifier in $[Mod]$, measuring the usefulness of the modifier for the given emotion. For the first 100 successful goal locations mod is chosen at random from $[Mod]$, to help avoid over-fitting to e and to share experience across the initial modifiers. After this, the m alternates between the highest weighted Emotion-Modifier connection and roulette wheel selection.

The most useful Emotion-Modifier connection $Weight_{em}$ is determined by the con_{em} and the fitness of the modifier F_m with the highest weighted connection being selected.

$$Weight_{em} = con_{em} + 0.10 * F_m \quad (5.6)$$

The aim of the weighting is to select modifiers with a strong connection to a specific emotion, but only if the modifier is considered to be fit. F_m is weighted by 0.1 to allow the connection between the modifier to be the dominate feature. If F_m has a weighting of one, the system would only ever choose the best modifier. Finally, once a modifier is selected the parameters of the navigation stack are adjusted. This process continues until the robot reaches the target location. Once the robot does reach the goal location a reward is provided and the system is updated.

5.3.4 Update

The emotion system is updated when a reward is provided by the environment, i.e. when the robot reaches the current goal location. The reward value is determined by an arbitrary value of 1000 plus the inverse time taken, in order to encourage the robot to complete goals as fast as possible.

$$reward = 1000 + 1000/time \quad (5.7)$$

However, if the robot reaches a state such that it must abort navigation, the system is updated with a reward value of 0. Aborting navigation is detected by the navigation stack and occurs if the robot is considered stuck or finding a valid path is not possible. Each connection is updated according to the update procedure with the relevant reward.

Reinforcer-Emotion update

The Reinforcer-Emotion connection is a standard XCS classifier and is updated using the process described in section 2.6. The higher the fitness of the classifier, the better the connection between the reinforcer state and the emotion category.

Emotion-Modifier update

The system updates each Emotion-Modifier mapping using the delta rule.

$$con_{em} = con_{em} + \beta * (Reward - con_{em}) \quad (5.8)$$

The learning rate β controls the rate at which the connection is learned, being commonly set to a value of 0.1. The higher the Emotion-Modifier weighting, the more useful the modifier is considered for that particular emotion state.

Modifier

A modifier's fitness is updated using the delta rule.

$$F_m = F_m + \beta * (Reward - F_m) \quad (5.9)$$

The higher the fitness value the more useful the modifier was for successful navigation.

During navigation the robot may utilise multiple Reinforcer-Emotion-Modifier connections, bow-ties, before reaching the goal location and receiving a reward, making it a multi-step problem. Therefore, the reward must be propagated to all the connections used to reach the goal location. It is not easily distinguishable which of these connections were useful, or detrimental to navigation. However, the most recently used connections can be considered to be useful for leading the robot to the goal location as they did successfully reach the goal. Therefore, the reward is propagated back to all connections with a discount factor (set to 0.5), giving maximum reward for the most recently used connection.

An algorithmic description of this process is given in algorithm 2. The next section will provide details on the experiments used to train and evaluate the emotion model described in this section.

$$Reward_i = Reward_{total} * discountFactor^{-i} \quad (5.10)$$

5.4 Experimental setup

The first set of experiments aims to train the emotion system on a simulated robotic platform. The second experiment investigates the ability of the system to learn useful bow-tie structures (termed the 'learn-ability' of the system). The third experiment evaluates the performance of the trained emotion model against the rigid navigation system, to discern any

Algorithm 2: Emotional learning algorithmic description

Data: $[bowties]$: bow-tie set; $[M]$: Match set; $[E]$: Emotion set;
 e : Emotion; $[Mod]$: Modifier set; mod : Modifier; R : Reinforcer

```

1 iteration = 0;
2 repeat
3   goal = getGoal();
4    $[bowties] = []$ ;
5   time = currentTime();
6   repeat
7      $R = \text{readNewReinforcerState}()$ ;
8      $[M] = \text{MatchSet}(R)$ ;
9     if  $[M].\text{size}() \leq \text{MIN\_MATCH\_SIZE}$  then
10      |  $\text{cover}([M], R)$ ;
11       $e = \text{SelectEmotion}([M])$ ;
12       $[E] = \text{emotionSet}([M])$ ;
13      if  $\text{TimeForGA}(E)$  then
14      |  $\text{applyGA}(E)$ ;
15      if  $[Mod].\text{size}() \leq \text{MOD\_SIZE}$  then
16      |  $\text{generateModifier}([Mod])$ ;
17       $m = \text{SelectModifier}([Mod], e)$ ;
18       $bowties += [[E], mod]$ ;
19       $\text{robot.applyModifier}(mod)$ ;
20    until Goal Discovered OR Goal Unreachable;
21    time = currentTime() - time;
22    if Goal Discovered then
23      |  $\text{rewardTotal} = 1000 + 1000/\text{time}$ ;
24    if Goal Unreachable then
25      |  $\text{rewardTotal} = 0$ ;
26     $\text{rewardConnections}(\text{rewardTotal})$ ;
27    if  $[Mod] \geq \text{MOD\_SIZE}$  AND  $\text{iteration}\%20 = 0$  then
28      |  $\text{generateModifier}([Mod])$ ;
29    iteration++;
30 until End of Training;

```

improvements to the robot's navigation performance. The final experiment tests the validity of the system on a real-world robotic platform.

The performance of the navigation systems is measured as: the number of collisions a robot has with the environment, and the time required for a robot to navigate to given goals. Ideally, the robot would have zero collisions with the environment and not require an excessive amount of time to reach the goal location.

From a psychological perspective, the emotion model provides an insight into the formation of emotions. The resulting bow-ties in each experiment will be examined for emergent emotions. The emotions are judged based on the relationship between the reinforcer input and the resulting behaviour of the robot's navigation.

5.4.1 Experiment One: Simulated emotion model training

In order to train the emotion system a Pioneer (as described in section 3.4.1) is tasked to navigate between goal locations within the simulated environment of the Willow Garage offices (seen in figure 5.6) using the Gazebo simulator (as described in section 2.8). The Willow Garage environment is an open access testing area of a large office environment, providing useful features to train the system with.

Placed into the environment are random placements of cupboards, office chairs, and desks to provide obstacles to navigate. Office chairs in particular are considered a difficult obstacle to tune a navigation algorithm to handle as the base of an office chair is generally larger than the stem of the chair. The sensors of a robot are often placed such that only the stem of the chair is visible, aliasing the size of the base of the chair which the robot will collide with. The size of an obstacle can be artificially increased via an inflation radius, however this has the disadvantage of narrowing the pathable area in tight doorways and reducing the effectiveness of navigating through them. Being able to adapt to each scenario would be advantageous.



Figure 5.6: Willow Garage offices in Gazebo simulator. All rooms are accessible, barring the top right room which is outdoors. Exits to the building were covered to prevent the robot leaving the office.

The emotion system is learned as the robot traverses from goal location to goal location, with reward being provided if the robot successfully reaches the current goal location. When the robot reaches a goal location a new goal location is randomly selected from the rooms in Willow Garage. If the robot must abort its traversal, the bow-tie is given a reward of zero, and the robot is reset to a random way point. To investigate the *learn-ability* of the emotion model, the number of emotion states is increased for each subsequent experiment, starting at two emotion states. One emotion is effectively the equivalent of learning a singular modifier, and does not provide adaptability to the system. The *learn-ability* of the system is defined as the ability of the system to learn a population of classifiers that are beneficial for path-planning. Investigating the number of emotions that can be learned all at once is important as the complexity of the bow-tie search space increases with the number of emotions, potentially becoming too complex to effectively learn with the XCS approach. Furthermore, this provides an insight into the number of emotions required for successful navigation for a given domain.

5.4.2 Experiment two: Learn-ability of the system

This experiment compares the bow-tie structure to a direct Reinforcer-Modifier model. This provides a benchmark to investigate the benefits of having the intermediate emotions. The hypothesis being that the emotions compact the search space and reduce the complexity (total number of state-action pairs) of the system, allowing XCS to learn beneficial bow-ties.

The Reinforcer-Modifier model removes the intermediary emotion link in the presented bow-tie model, with classifiers directly matching a single modifier in a typical Condition-Action set-up. The Reinforcer-Modifier model will be trained as described in Emotion training, see section 5.4.1.

5.4.3 Experiment three: Emotion model path-planning performance

The robot is tasked to navigate a course in Willow Garage to test if the learned emotion system has any benefit to navigation. To judge the effectiveness of the navigation, the time taken to navigate and the number of collisions will be measured. These results are then compared to the default static navigation stack parameters.

After emotion training is complete, the bow-tie structure is made read-only. A set of five way points is set up in the Willow Garage simulation, seen in figure 5.7. The Pioneer is tasked with completing the course in order using the default navigation system, and the learned emotion system. Although the environment and learned system are not stochastic (no further learning occurs at the testing stage) the test is repeated 10 times as simulated noise may effect performance.

Each path has its own particular challenges for the robot to navigate. Path one requires navigating through a narrow door. Path two is a series of tight spaces. Path three is a wide space with large doorways. Path four is a long narrow corridor. Path five is characterised by small doors within a narrow corridor.

5.4.4 Experiment four: Real-world emotion model training

In order to test the validity of the emotion learning in the real world, the emotion training (section 5.4.1) and the benchmark (section 5.4.3) testing is repeated in the real-world. The system is trained on the Pioneer robot in a real-world office environment described in section 3.4.2 and shown in figure 3.6 on page 69. The real-world course the robot will be tested on is shown in figure 5.8 on page 127.

Running 30 trials for statistical analysis on the real-world robot is prohibitively time consuming. However, a single trial still provides an insight into the learn-ability of the emotion system for real-world robotics.

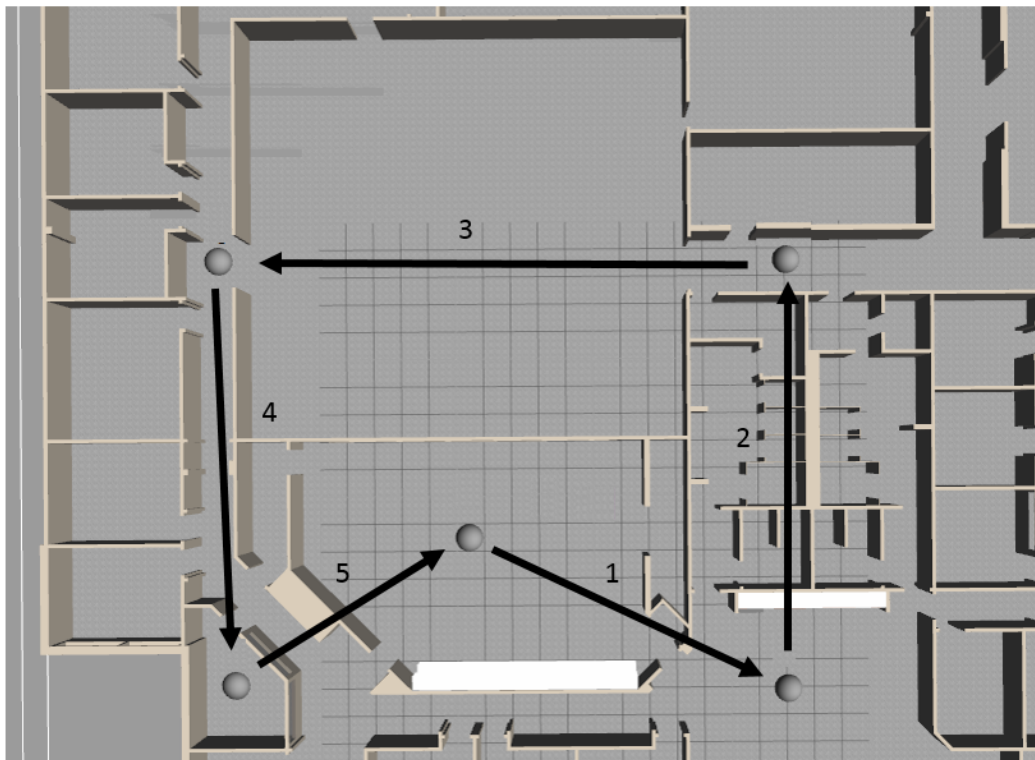


Figure 5.7: Simulated benchmark course, each ball indicates a way-point in the course around the Willow Garage office building. The arrows indicate the path order that the robot traverses in order to reach the goal location, albeit doors, open rooms, and corridors must be used to navigate.

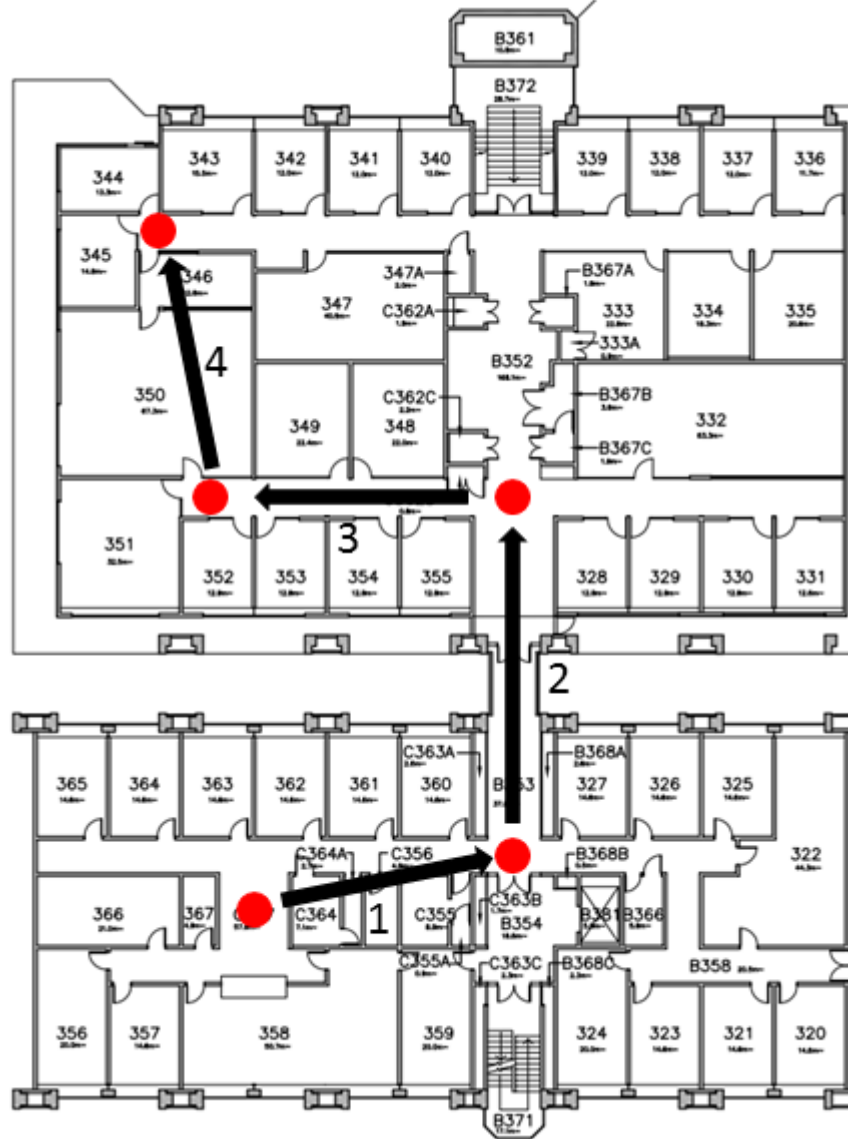


Figure 5.8: Real-world benchmark course, each ball indicates a way point in the course around the Cotton building level three. The arrows indicate the path order that the robot traverses in order to reach the goal location.

Furthermore, this experiment provides a comparison with the simulation results.

Chapter 6

Emotion inspired adaptive robotic path planning results

The previous chapter detailed the emotion model. Through learned emotional responses, the emotion model adapts a robot's path-planning cognition as it interacts with the environment. The hypothesis is that these learned emotional responses will improve the robot's navigation performance by providing the robot with the flexibility to change its behaviour based on its interaction with the environment.

This chapter first analyses the results of training the emotion model from a psychological perspective regarding the emotional responses that may have emerged in the resulting bow-ties. Second, a discussion on the learn-ability of the emotion model is given, including a comparison against a learning system with direct Reinforcer-Modifier mappings (i.e. infinite emotions). Finally, this chapter provides results evaluating the navigation performance of the emotion system by comparing it against the rigid path-planning approach.

The initial experiments reported in this chapter trained and tested the emotion model within the simulated Willow garage environment (see figure 5.6 on page 123). This was because XCS can take a large number of learning iterations to learn a population of solutions, requiring long train-

ing times. The simulator provided a means of running 30 training runs for repeatability testing and statistical analysis. To validate the simulated results, the final experiment ran a single training run on the real-world Pioneer. This single real-world training run took 14 days to complete (over a four week period), compared to an average of seven days to complete 30 simulated training runs.

6.1 Experiment One: Simulated emotion model training

The aim of this experiment was to train the emotion model and investigate the emergence of emotions within the bow-ties. To train the emotion model, a simulated Pioneer was tasked to navigate between goal locations within the Willow Garage offices (as described in section 5.4.1). The Pioneer had to successfully navigate to a goal location 1000 times before the training of the emotion model was halted. This was then repeated 30 times to investigate variance.

To determine if the emotion model was repeatedly learning emotional responses, the classifier populations in each training run were combined together into a single population. This was done by clustering classifiers based on the modifier it elicited. This is because one training run can associate a bow-tie to emotion state one, while another training run may have associated the same bow-tie to emotion state two. But both still elicit the same behaviour, the emotion states are merely labelled differently due to the learning process. An example of two similar (lowest absolute difference between values) modifiers is shown in table 6.1.

Once the classifiers are clustered into their respective emotion states, the reinforcer conditions of the classifiers are examined to determine if there are any unique patterns that elicit that particular emotion state. For example, one emotion may be elicited by classifiers that have high novelty values, while another emotion is elicited by classifiers with low novelty

Table 6.1: Example of two emotions with different labels that would be clustered into a single emotion state.

Emotion	Inflation radius	Cost scaling	Speed	Occlude distance	LIDAR	Sonar	Collision
One	0.85	5.98	2.99	0.02	0.81	0.40	0.32
Two	0.77	6.28	2.49	0.02	0.74	0.44	0.52

values.

In order to investigate the learn-ability of the emotion model, at the end of the 30 training runs the number of emotion states was increased and the emotion model was re-trained from scratch (initial training starting with two states).

6.1.1 Two Emotions

Inspection of the results shows a trend for two emotion states to emerge with similar modifier parameters, as seen in table 6.2. The values of the modifiers are significantly different between each modifier, indicating the emotion model is repeatedly learning two distinct behaviours. Two modifiers were considered significantly different if one of their parameters passed a two-tailed student T-Test at 95% confidence.

These two distinct parameter settings can be seen in figure 6.1 showing the grid weightings for each modifier. Emotion one has a relatively lower weighting for *obstacles* causing it to be less restrictive on the movements of the robot platform compared with emotion two. Conversely emotion two restricts the robot's path options near obstacles, in particular around doorways where low cost paths can be seen tightly down the centre of the doorway. Furthermore, emotion one tends to increase the speed of the robot compared to emotion two.

By inspection, there is a pattern of fit Reinforcer-Emotion-Modifier classifiers in each population that matches one of the two emotion states, examples of which are shown in table 6.3. These classifiers consistently have relatively high fitness, in the top five of the population, and are consistent

Table 6.2: Learned most fit emotion to modifier mapping averaged over 30 trials: two emotions

Emotion	Inflation radius	Cost scaling	Speed	Occlude distance	LIDAR	Sonar	Collision
One	0.85 ± 0.20	5.98 ± 1.30	2.99 ± 0.90	0.02 ± 0.01	0.81 ± 0.22	0.40 ± 0.11	0.32 ± 0.11
Two	7.62 ± 1.20	0.62 ± 1.2	1.06 ± 0.10	0.02 ± 0.02	0.75 ± 0.12	0.36 ± 0.13	0.85 ± 0.21

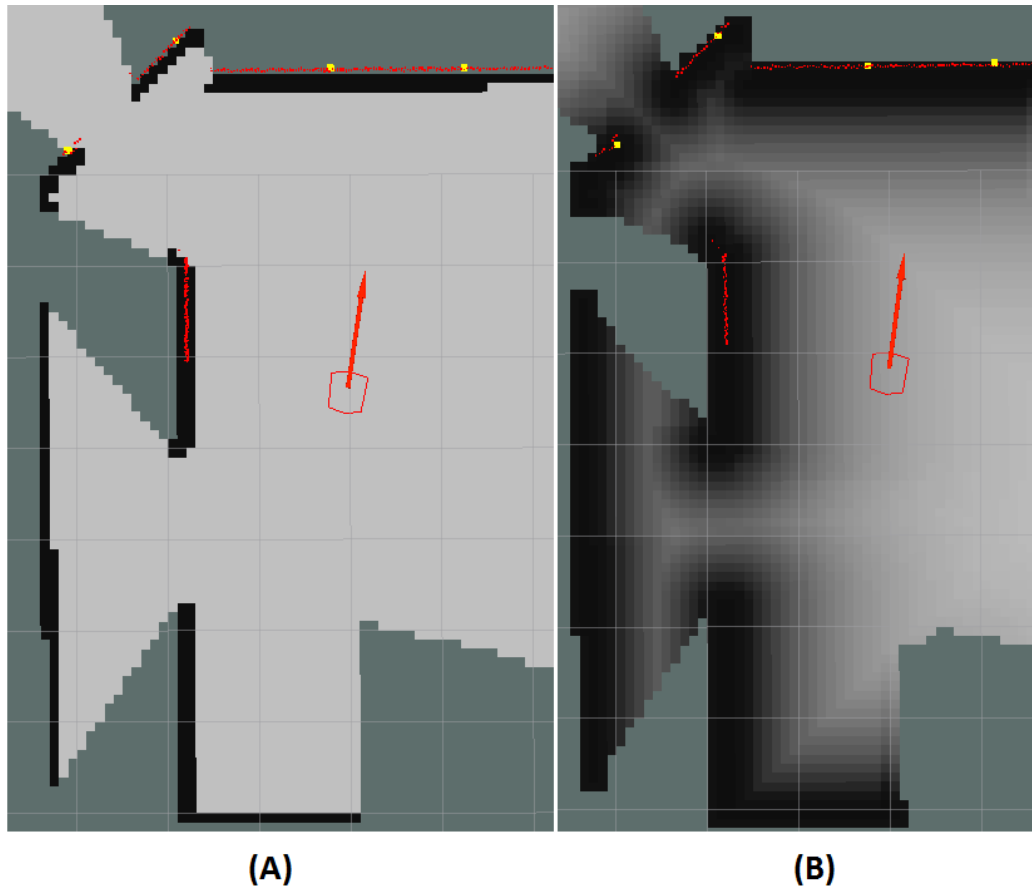


Figure 6.1: (A) Behavioural response of emotion one (“happiness”), note how the occupancy grid is mostly light grey in colour indicating low cost of path planning in most areas. (B) Behavioural response of emotion two (“fearful”), note how the occupancy grid is darker grey in colour, compared to (A), indicating a higher cost to path planning reducing suitable regions for navigating.

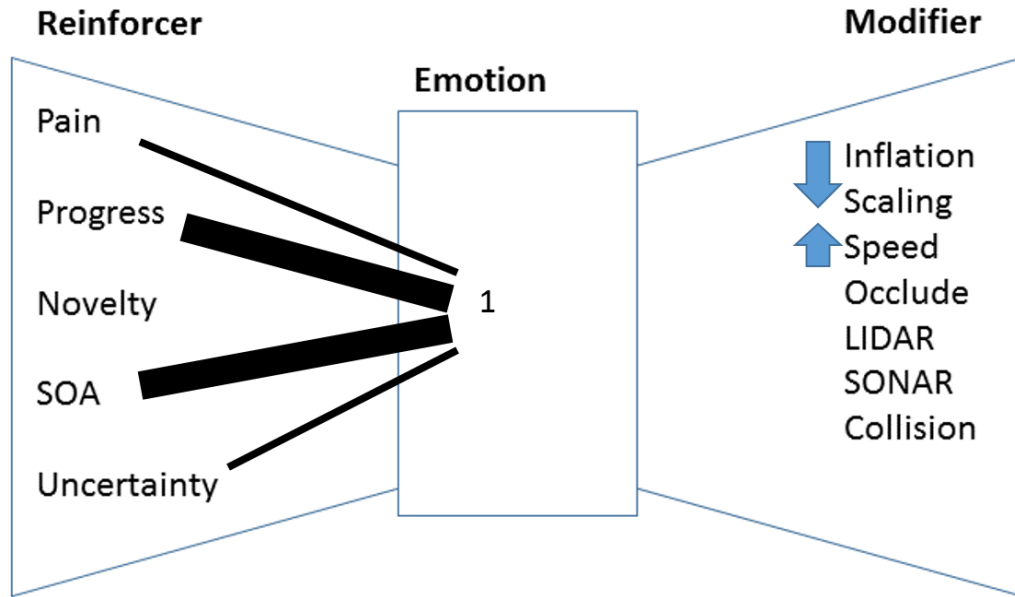


Figure 6.2: Bow-tie learned with two emotion states for emotion one

(between training runs) in the emotion they elicit.

Classifiers with low to no pain reinforcement or high progress tend to select emotion one (using the normalised emotion labels as above). Emotion two tends to be selected by reinforcement with either pain, or low progress and high novelty. A visual representation of these bow-ties is given in figures 6.2, 6.3, and 6.4. In each figure the thickness of the line between the Reinforcer and Emotion is the value's weight for selecting the given emotion, no line indicates a 'don't care' value. The arrows then represent the strength of the modifier values, the thicker the line the larger the reinforcer value. The arrows in the modifier values indicate an increase or decrease in value compared with the default navigation stack values.

Considering the results of the emotion to modifier learning in table 6.2, two distinct behaviours for each emotion can be seen. Classifiers that include emotion one have reinforcers with low pain or high progress. Emotion one then elicits a modifier that lowers the cost to navigate and increases the speed of the robot. The system has effectively learned to de-

Table 6.3: Example of the consistent Reinforcer-Emotion classifiers for the emotion model trained with two emotion states

Emotion One						
	Proportional		Integral		Differential	
	Lower	Upper	Lower	Upper	Lower	Upper
Pain	#	#	-1.00	-0.60	#	#
Progress	#	#	0.60	1.00	#	#
Novelty	#	#	#	#	#	#
SOA	#	#	0.40	0.80	#	#
Uncertainty	#	#	-0.60	-0.20	#	#
Emotion Two						
Pain	#	#	#	#	#	#
Progress	#	#	0.00	0.40	#	#
Novelty	0.00	0.40	0.60	1.00	#	#
SOA	#	#	#	#	#	#
Uncertainty	#	#	#	#	#	#
Emotion Two						
Pain	#	#	0.5	1.00	#	#
Progress	#	#	#	#	#	#
Novelty	#	#	#	#	#	#
SOA	#	#	#	#	#	#
Uncertainty	#	#	#	#	#	#

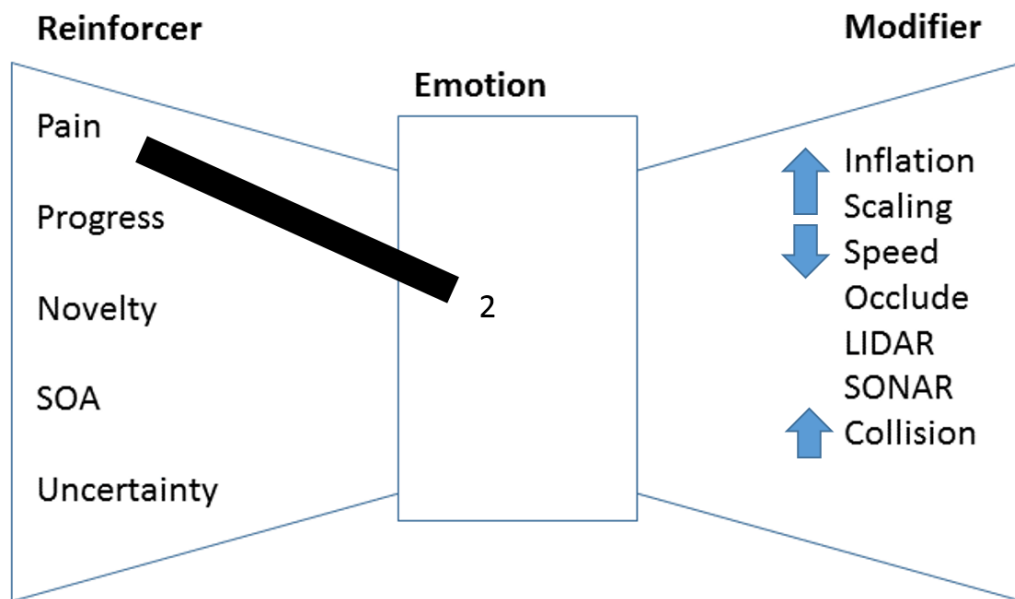


Figure 6.3: Bow-tie learned with two emotion states for emotion two

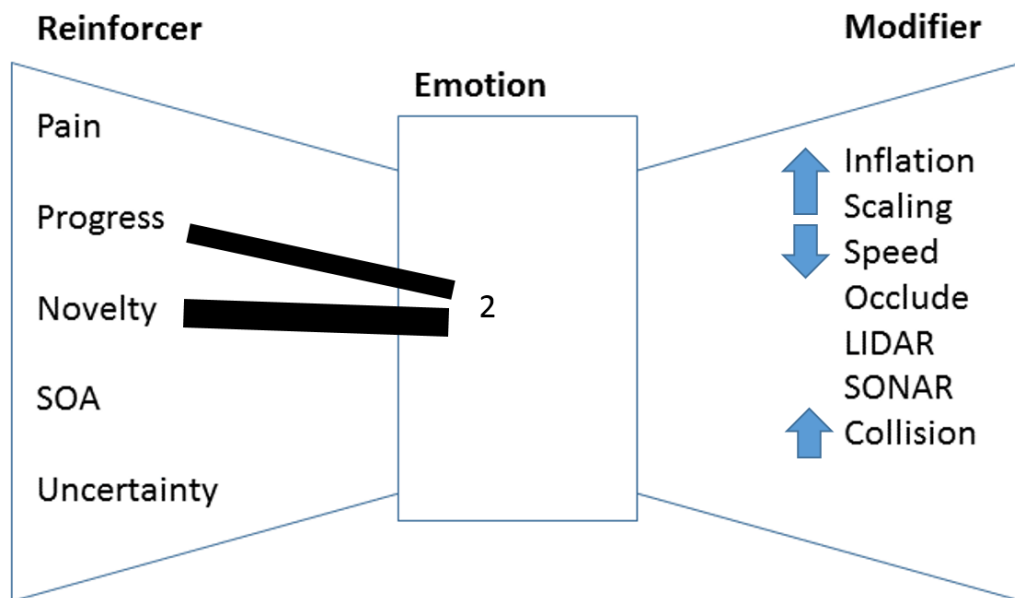


Figure 6.4: Bow-tie learned with two emotion states for emotion two

crease the safety margins around obstacles in the environment when the robot is not colliding with the environment. This allows the robot to drive faster and potentially move closer to objects within the environment.

Classifiers that include emotion two have reinforcers with high pain, low progress, or high novelty. Emotion two then elicits a modifier that increases the cost to navigate, and lowers the speed of the robot. The system has learned to increase the safety margins around obstacles in the environment when the robot is colliding with the environment (experiencing pain) or is within a novel area in the environment. This restricts the path the robot will follow near obstacles, effectively directing the robot away from obstacles to potentially prevent collisions.

An anthropomorphic description of emotion one can be considered to be ‘happiness’, as described in 5.2. Often triggered by goal achievement, the receipt of a reward, or the absence of an expected punishment, happiness encourages an animal or human to repeat rewarding (or punishment-avoiding) behaviours. Positive emotions associated with happiness have also been linked to a broadening of cognitive focus [47], directing attention outward and encouraging interaction with the environment [9]. This is seen by the lowering of safety margins, allowing the robot to plan closer to obstacles. An anthropomorphic description of emotion two can be considered to be ‘fear’. Fear often occurs when a punishment is expected. It increases risk-aversion and encourages the avoidance of, or escape from, dangerous stimuli [97].

6.1.2 Three Emotions

The emotion system’s ability to construct emotions was increased to three emotion states. The results of the learned modifiers are shown in table 6.4. Of note in the results is that two of the Emotions (One and Two) have similar modifiers to those learned in the two emotion experiment. Emotion one reduces safety margins, while emotion two increases safety margins. The new emotion (three) has modifier values in-between the modifier val-

Table 6.4: Learned most fit emotion to modifier mapping averaged over 30 trials: three emotions

Emotion	Inflation radius	Cost scaling	Speed	Occlude distance	LIDAR	Sonar	Collision
One	0.75 ± 0.10	6.8 ± 1.10	3.1 ± 0.78	0.02 ± 0.06	0.79 ± 0.22	0.48 ± 0.16	0.32 ± 0.13
Two	7.62 ± 1.20	0.62 ± 1.60	1.06 ± 0.10	0.01 ± 0.02	0.74 ± 0.22	0.44 ± 0.16	0.85 ± 0.21
Three	4.62 ± 2.20	1.10 ± 1.05	1.68 ± 0.20	0.01 ± 0.01	0.82 ± 0.12	0.50 ± 0.13	0.55 ± 0.11

ues of emotions one and two. Emotion one and two are contrasted by the relatively extreme difference in the cost values of obstacles, with low and high levels of inflation for each emotion respectively. Emotion three's modifier is in-between the two with a medium level of inflation, preventing the robot from getting close to walls but not driving the robot away from walls.

By inspection, each of the three emotions has a common classifier associated with it, examples of common high fitness classifiers are shown in table 6.5. Classifiers that choose emotions one and two are similar to the previous results. Emotion one is selected by classifiers with low to no pain, or high progress. Emotion two is selected by classifiers with high pain, or no progress. The new emotion state three is selected by classifiers with relatively medium levels of proportional pain, between the first two emotions proportional pain values.

Emotion three increases safety margins when the robot has been colliding with the environment. Emotion three will plan paths further from obstacles if the robot collides with the environment but not to the extreme extent that emotion two will. In effect, emotion three is a more cautious path-planning approach to emotion one but not as overly restrictive as emotion two is.

Taking into consideration the Reinforcer-Modifier-Emotion connections the anthropomorphic descriptions of emotions one and two can again be judged as happiness and fear. Emotion three however, appears to be a middle ground between the first two emotions. Emotion state three is increasing aversion to obstacles, and slowing speeds but not to the levels of

Table 6.5: Example of the consistent Reinforcer-Emotion classifiers for the emotion model trained with three emotion states

Emotion One						
	Proportional		Integral		Differential	
	Lower	Upper	Lower	Upper	Lower	Upper
Pain	#	#	-0.80	-0.40	#	#
Progress	#	#	0.50	0.90	#	#
Novelty	#	#	#	#	#	#
SOA	#	#	#	#	#	#
Uncertainty	#	#	-0.80	-0.40	#	#
Emotion Two						
Pain	#	#	0.40	0.80	#	#
Progress	#	#	0.00	0.40	#	#
Novelty	#	#	#	#	#	#
SOA	#	#	#	#	#	#
Uncertainty	#	#	#	#	#	#
Emotion Three						
Pain	#	#	-0.10	0.10	#	#
Progress	#	#	#	#	#	#
Novelty	#	#	#	#	#	#
SOA	#	#	#	#	#	#
Uncertainty	#	#	-0.90	-0.50	#	#

Table 6.6: Learned most fit emotion to modifier mapping averaged over 30 trials: four emotions

Emotion	inflation radius	cost scaling	speed	Occlude distance	LIDAR	sonar	collision
One	2.75 ± 2.80	6.8 ± 2.10	2.1 ± 1.78	0.02 ± 0.06	0.90 ± 0.10	0.48 ± 0.16	0.72 ± 0.23
Two	4.62 ± 4.20	1.10 ± 2.05	1.68 ± 1.20	0.01 ± 0.01	0.82 ± 0.12	0.50 ± 0.13	0.62 ± 0.41
Three	4.62 ± 5.20	1.10 ± 3.05	2.38 ± 1.40	0.01 ± 0.01	0.72 ± 0.14	0.50 ± 0.13	0.53 ± 0.19
Four	4.62 ± 2.20	1.10 ± 2.05	3.20 ± 1.50	0.01 ± 0.01	0.84 ± 0.16	0.50 ± 0.13	0.85 ± 0.11

fear which drastically increases the aversion to walls with two emotional states available. Based on Rolls' scale of emotions, shown in figure 2.4 on page 27, this could be considered as apprehension. Apprehension being the anxiety or fear that something bad or unpleasant may happen. In this case of robotic navigation, the robot has encountered a small amount of pain and is increasing safety margins in response.

6.1.3 Four Emotions

The results of extending the emotion model to four emotions does not show a clear pattern in either the modifier or classifier populations. The modifier population, shown in table 6.6, shows non-distinct values between emotions, indicated through the large variance in each of the clustered Emotion-Modifier responses. The classifier population does not show any clear Reinforcer-Emotion-Modifier patterns either. The most fit classifiers are also lowly weighted, with an average fitness in the top five classifiers of 0.14, compared with 0.65 and 0.42, for two and three emotion states respectively. This indicates the system was not learning reliable or useful bow-ties connections, and is not expected to improve performance and may even potentially hinder navigation. The potential cause of this result is discussed in the next section, where the results are compared against a direct Reinforcer-Modifier model.

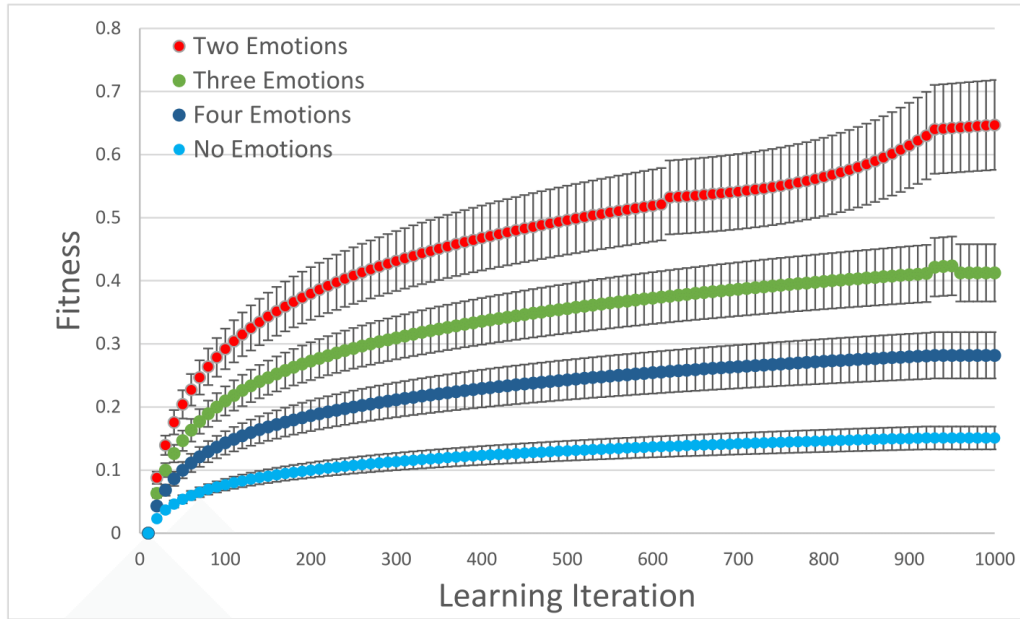


Figure 6.5: Average classifier fitness for model with zero to four emotions

6.2 Experiment two: Learn-ability of the system

In order to get an insight into the learn-ability of the emotion model, the fitness of the classifier and modifier populations is examined. Figure 6.5 shows the average classifier fitness for the top five classifiers in the population, and figure 6.6 presents the average fitness for the top five modifiers. The higher the fitness of the classifier the more reliable the classifier is for successful navigation.

Comparing both figures it can be seen that as the number of emotions is increased, the fitness of both the modifier and classifier populations both decrease. Similarly, the initial rate at which the fitness increases also decreases. The system with no emotions (direct Reinforcer-Modifier connections) is the worst performing system with fitness in both populations being significantly lower than the emotion models. In each of the systems the classifier fitness has plateaued in growth during the 1000 (successful)

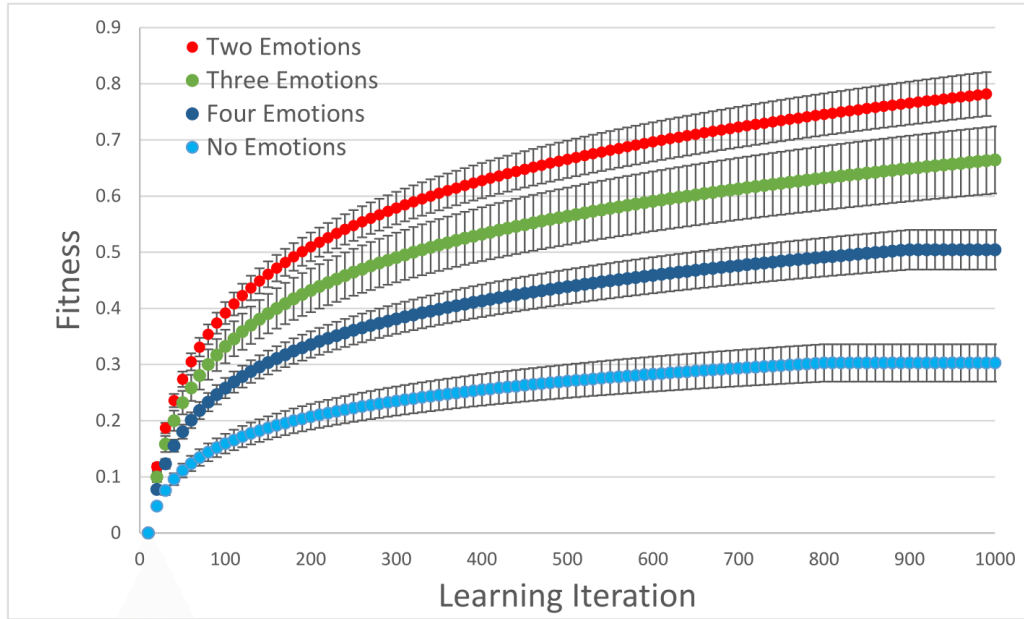


Figure 6.6: Average modifier fitness for model with zero to four emotions

navigation runs. In the two and three emotion models the modifier behaviours appear to still be increasing in fitness at a relatively minor rate.

Another trend is that the modifier fitness rises ahead of the classifier fitness, which is observed as the modifier fitness increasing ahead of the classifier fitness in figures 6.5 and 6.6. This is reasonable as a classifier's fitness is dependant on the modifier population having useful behaviours for the system to connect emotions to. The classifiers will not reliably receive reward until the system learns useful modifiers (modifiers that allow the robot to reach the goal locations) for the emotions to consistently match.

The drop in modifier fitness as the the number of emotions increases may be due to the amount of exploration required when matching emotions to the modifier set. The system is required to match each emotion to a modifier value. Increasing the number of emotions increases the search space of Emotion-Modifier connections. In the two emotion system, the model is searching for two reliable modifiers. As this number of emotions

increases, the system is required to find more modifiers (assuming there are more beneficial modifiers to find). Increasing the search space has potentially limited the amount of reward each modifier can receive over time, as the chance to be selected is lower.

This is seen in the extreme case where no emotions are present for filtering. The no emotion system has to directly match Reinforcer input to the Modifiers, learning a theoretically infinite number of emotion states. The system has failed to learn the complex web of inputs. From this it can be concluded that having two or three emotions as filters has allowed the system to reduce this complex web and learn emotion states which have reduced the number of collisions and time to navigate. The next section will now investigate how these results translate into the real world.

6.3 Experiment three: Emotion model path-planning performance

The previous section has presented the results of learning the bow-ties. This experiment evaluates the navigation performance of the emotion model against the rigid path-planning approach. This will be measured by comparing the navigation time, and number of collisions as the robot traverses a given path of the simulated environment, as described in section 5.4.3. Ideally the robot would never collide with the environment, as collisions risk damage to the robot or the environment. However, it is also desirable that the robot still travels as quickly as it can, without causing collisions.

Figure 6.7 shows the time to navigate to each check point by the learned emotion models compared with the default navigation stack values. Figure 6.8 on page 144 shows the number of collisions during navigation for each approach. The two emotion bow-tie is equal to or out-performs the default values in both time and collisions, in particular for the most challenging path, i.e. checkpoint one to two which has the narrowest corridors with the highest risk of collision. The two emotions also provide a

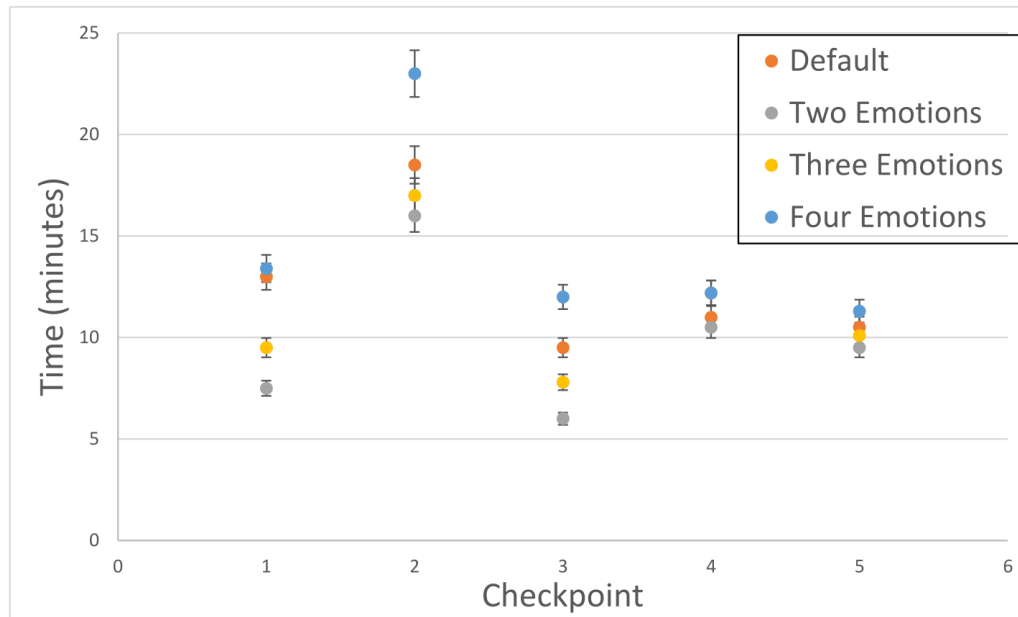


Figure 6.7: Simulated comparison of time between each checkpoint

time improvement in the open spaces between checkpoints one and three. The three emotion bow-tie provides similar improvements over time and collisions to the two emotion bow-tie. The time improvement is not as significant an improvement as the two emotion bow-tie, but does provide a benefit to navigation. The four emotion bow-tie however, has performed worse than the default settings, both increasing time to navigate and the number of collisions. This indicates that the four emotions did not learn a beneficial bow-tie.

The primary advantage demonstrated by the emotion system is the ability to adapt the robot's speed and safety margins based on its reinforcer state, which can be seen prominently in time and collision improvements in paths two and three. When navigating narrow regions, robots can tend to collide with obstacles more often, with the 'fear' response causing the robot to increase its obstacle aversion that restricts paths from being near obstacles. In a static path planning approach this would adversely effect

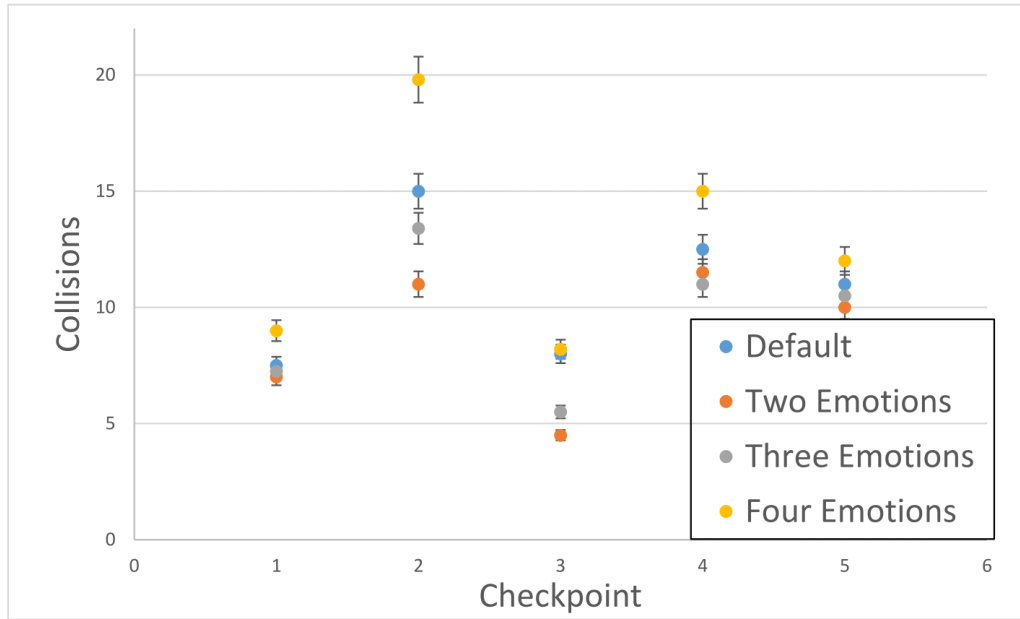


Figure 6.8: Simulated comparison of collisions between each checkpoint

performance in wide spaces, resulting in the robot going unnecessarily slow. However, the happiness response causes the robot to increase its speed when collisions are low, providing a faster behaviour in open areas. Although similar behaviour can be pre-programmed, this behaviour is emergent from the novel learning system.

It was possible that more emotions, up to an undetermined size, would be more beneficial by providing a larger range of behaviours. These results indicate that two emotions is the more beneficial option, with three emotions being relatively similar in performance. The next section will analyse the fitness of the classifier and modifier populations to investigate how increasing the number of emotions effects learning.

Table 6.7: Learned most fit emotion to modifier mapping: real-world experiment

Emotion	Inflation radius	Cost scaling	Speed	Occlude distance	LIDAR	Sonar	Collision
One	1.2	4.98	2.50	0.02	0.91	0.80	0.72
Two	3.62	1.50	1.54	0.02	0.82	0.41	0.85

6.4 Experiment four: Real-world emotion model training

The real-world experiment will investigate how and what the emotion model learns in a real-world environment. Given the time taken to learn the emotion model in a simulated environment, the real world training is restricted to two emotions and only one complete training run of 1000 successful runs. The overall time to complete this experiment was 14 days, compared to the average seven days to complete 30 training iterations in simulation.

Inspection of the results shows a familiar trend with two distinct emotion states emerging, shown in table 6.7. Similar to the simulated results, emotion one has a lower weighting for obstacles and is less restrictive on the movements of the robot platform compared to emotion two. However, each emotion does not effect change on the safety margins to the same extreme as the simulated results.

Inspecting the classifier population reveals that there are two common high fitness classifiers for each emotion state. Emotion one is elicited by classifiers with no pain, high progress, and low uncertainty. This is the same as the simulated results, which was considered as an emotional response of happiness. Emotion two is elicited by classifiers with pain, similar to the simulated results for emotion two. However, only pain is considered in the real-world results compared to the simulated emotion two which also considered novelty and uncertainty. As the real-world environment is smaller than the simulated environment the novelty factor is lower, and the robot's position estimate should vary less. This could

Table 6.8: Example of the consistent Reinforcer-Emotion classifiers for the emotion model trained with two emotion states: real-world

Emotion One						
	Proportional		Integral		Differential	
	Lower	Upper	Lower	Upper	Lower	Upper
Pain	#	#	-1.00	-0.60	#	#
Progress	#	#	0.20	0.60	#	#
Novelty	#	#	#	#	#	#
SOA	#	#	#	#	#	#
Uncertainty	#	#	-1.00	-0.60	#	#
Emotion Two						
Pain	0.60	1.00	0.20	0.60	#	#
Progress	#	#	#	#	#	#
Novelty	#	#	#	#	#	#
SOA	#	#	#	#	#	#
Uncertainty	#	#	#	#	#	#

have made the uncertainty and novelty less important for navigation in the real-world environment. Emotion two may still be considered as a ‘fear’ response, with pain eliciting an aversion to obstacles.

The navigation performance of the emotion model learned in the real-world was then measured as discussed in section 5.4.4. For comparison the bow-ties learned in simulation were also run in the real-world navigation test. The time and collision results are shown in figures 6.9 and 6.10 on page 148 respectively. Both bow-ties learned in simulation showed improvement in time and number of collisions compared with the default system. This provides a demonstration that the emotion models learned in simulation are applicable to real-world path planning.

The bow-ties learned in the real-world also show an improvement in

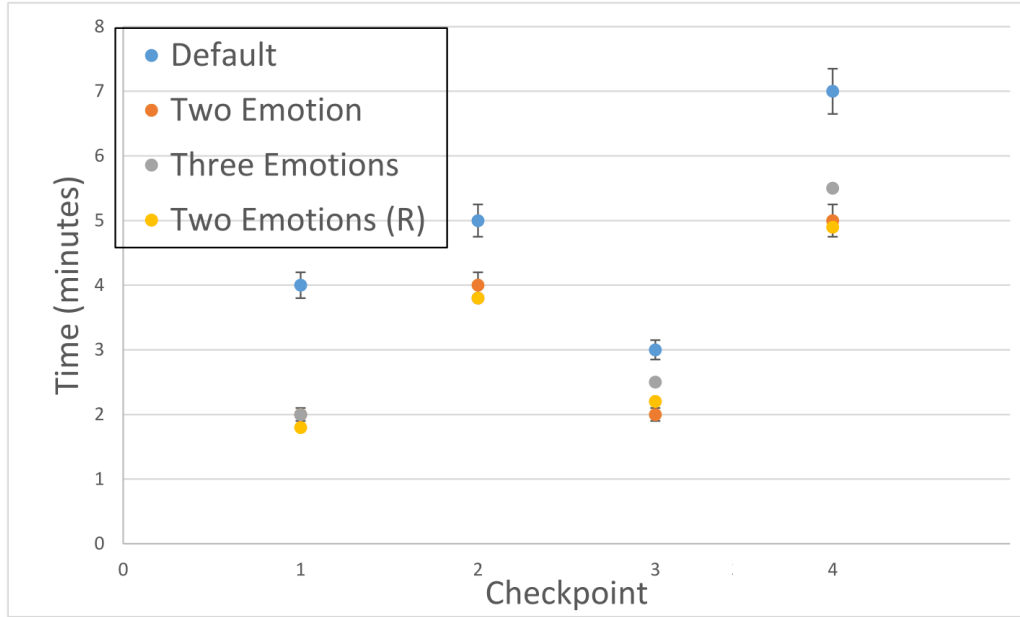


Figure 6.9: Real-world (R) comparison of time between each checkpoint

time performance and the number of collisions compared to the default system. This is a positive result demonstrating the system trained in the real-world can learn a beneficial bow-tie for robotic path planning. Comparing the real-world results to the simulated results, both of the two emotion systems outperform their three emotion counterparts, although the result is not significant (under the T-test) it is a similar result to the simulated tests. This supports the proposition that two emotions are the preferred number of emotions for path planning.

In order to get an insight into the learn-ability of the system in the real-world, the fitness of the classifier and modifier populations are illustrated in figure 6.11 on page 149. The figure shows that the classifier fitness again lags behind the modifier fitness, the same as the simulated results. The overall fitness of each population both rises at a lower rate and plateaus at a lower value than the simulated tests. This indicates that the real-world robot is not as successful at reaching way-points as the sim-

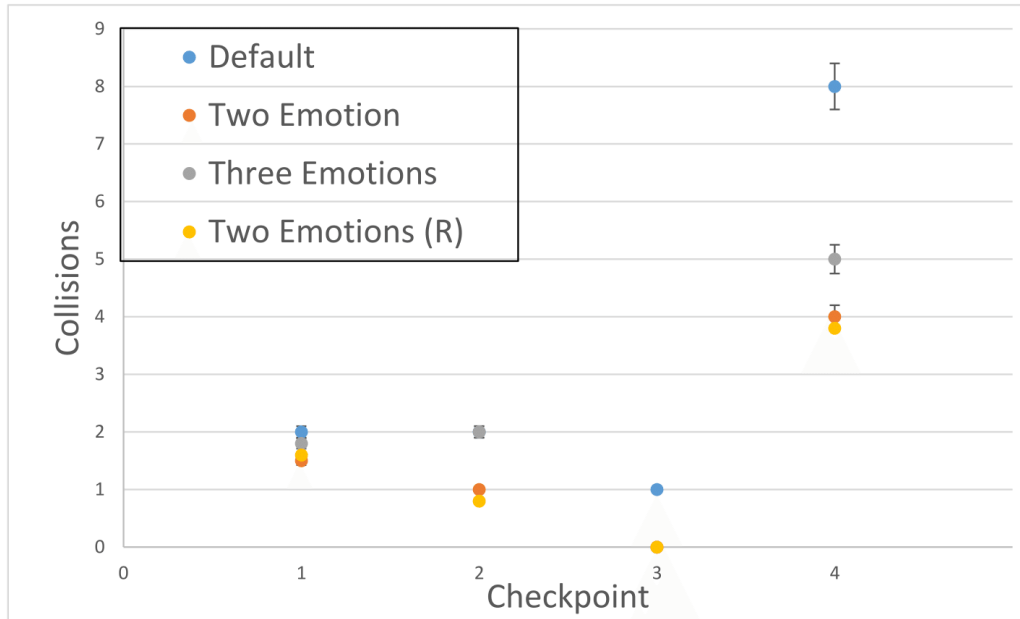


Figure 6.10: Real-world (R) comparison of collisions between each checkpoint

ulated robot. This is potentially due to the differences in simulation and real-world, such as sensor noise. Overall, the emotion system has learned similar emotion responses in the real-world and simulated training.

6.5 Discussion

Results from the simulated experiments have shown three human judged anthropomorphic emotion states; fear, apprehension, and happiness. These emotions have distinct Reinforcer-Emotion-Modifier bow-ties which define the behaviour of the robot platform based on its interaction with the environment. Each of the behaviours is an intuitive response; with increases in pain (collisions) from the environment, the robot should increase safety margins and slow speeds. Conversely the lack of pain should allow for exploration and potential risk taking, i.e. navigating closer to obstacles increasing the risk of collisions. Results in the real-world have

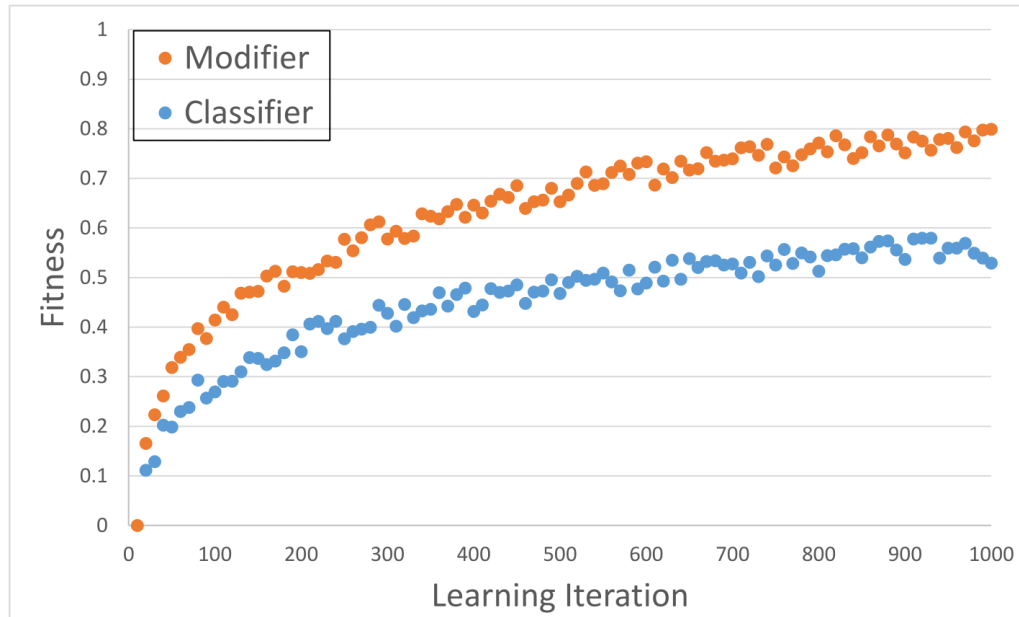


Figure 6.11: Real-world (R) comparison of classifier and modifier fitness

shown similar results with fear and happiness responses being learned, indicating the validity of the simulated results. Extending the system to four emotions however, did not show any clear signs of anthropomorphic emotions, and fitness values indicate the system was unable to learn.

A common trend in the learned classifier populations is the omission of the differential values being used by fit classifiers. This suggests that the differential term, i.e. the difference between current and previous value, is not useful for navigation or has not been useful for differentiating classifiers. From examining low fitness classifiers the differential term is most commonly zero or near to zero for most parameters, as each parameter in the system does not drastically change at each time step. The one exception is during a collision and the pain value will spike to one.

Because of this, subsumption has eliminated the specific classifiers with the zero values in the differential terms for classifiers with 'don't care' values as they have been considered more general. Subsumption is the pro-

cess of eliminating classifiers which do not add anything to the decision making process of the system. Subsumption works by attempting to compact the rules through removing overly specific classifiers, i.e. it makes the classifier population as generalised as possible. This indicates that the differential term is not useful for robotic navigation.

The differential term provides information on the change in reinforcer value, this is potentially useful for allowing a robot to learn to anticipate the next emotion or event (collision or power loss). Anticipation may be more useful in a dynamic environment or problem domain where the ability to anticipate the next state is important (anticipating a lack of energy or collision).

Considering the fitness evaluations, a classifier's fitness is dependant on there existing useful, high fitness, modifiers. However, increasing the number of emotions has a detrimental effect on learning the modifiers, seen by decreasing fitness values. A potential solution would be to divide the learning process between the modifier population and the classifier population. First learn a series of useful modifiers, and then learn reinforcement values to select when to use them. The limitation of this approach is in how the modifiers will be trained, without an environmental reference it is not clear how the system will select which modifier to test. From an evolutionary point of view, the emotional reinforcer is a necessary part of learning emotional behaviours.

According to modern evolutionary theory, different emotions evolved at different times [40]. Primal emotions, such as fear, are associated with ancient parts of the brain and presumably evolved among our earlier ancestors. Affectionate emotions, such as love for an offspring, seem to have evolved among early mammals. Social emotions, such as guilt and pride, evolved among social primates. Furthermore, more recently evolved parts of the brain can affect older parts of the brain, such as when the cortex moderates the amygdala's fear response. This means that an earlier emotion response or reinforcer may now be affected by more evolved cogni-

tive process. For example, an initial spider bite on a person's leg might not cognitively be noticed (pain response is too low). But then the toxin in the bite leads to extreme pain being felt, indicating that something is seriously wrong. The brain may then cognitively associate that initial minor pain sensation to the subsequent pain response, meaning the next time your leg brushes against something your brain cognitively increases the pain sensation you feel. This has led some researchers to propose that pain is an emotion response [19], meaning pain as considered in this work would be more accurately defined as a perception of physical feeling (touch).

Considering evolutionary theory, the navigation task the robot was required to complete is similar to the challenges faced by our early ancestors. There is no social interaction or community of robots, it would not be reasonable to consider the emotion model to learn emotions such as guilt, pride, or love in this case. Thus, the navigation task may only require primitive behaviours such as fear. This proposition is supported by the emotion model only producing 'fear' and 'happiness' responses. To learn more complex emotion behaviours within the presented emotion model the robot may be required to learn in social or co-operative tasks.

6.6 Summary

In summary, a human inspired emotion model has been trained to adapt the path-planning behaviour of a robot. Algorithmically the system has learned three policies capable of adapting the robot's navigation behaviour. These policies demonstrate three emergent emotional behaviours: fear, apprehension, and happiness. The results have demonstrated that these learned emotion responses have improved navigation performance of the Pioneer, when compared to the default non-adapting navigation system. This improvement is seen as the robot reducing the number of collisions with the environment and improving its time to navigate given paths. The results have also shown that two emotions improved the performance the

most, with more emotion states reducing the navigation performance due to reducing the learn-ability of the emotion model.

Chapter 7

Conclusions

Robotic navigation has traditionally been approached with rigid computational techniques, relying on pre-determined models to determine a robot's behaviour. The objective of the work presented here was to apply human inspired mechanisms to address limitations in current robotic navigation techniques. With specific emphasis on the problems of 'where am I?' (SLAM) and 'how do I get there?' (path-planning). In order to achieve this, two novel human inspired cognitive systems have been introduced that add flexibility to current navigation techniques by allowing a robot to learn through its interaction with the environment.

7.1 Cognitive action mapping

Teaching, alongside imitation, is widely thought to underlie much of the success of humanity by allowing high-fidelity transmission of information, skills, and technology between individuals, facilitating both cumulative knowledge gain and normative culture [37]. Inspired by this concept, this thesis has introduced a novel cognitive mapping stage to the SLAM process called CAM-SLAM with the objective of providing a means for robots to share navigation information.

CAM-SLAM is based on the concept within cognitive science of a sep-

arate cognitive and mental map in humans. The cognitive mapping stage functions in conjunction with current SLAM techniques, which are considered to be the mental mapping stage. The mental stage learns a detailed sensory map of the environment and provides localisation information. The cognitive mapping stage utilises an ACS to learn a series of novel *decision points* with associated *action* features linking them together in order to produce a cognitive action map (CAM). The key contribution is the utilisation of the decision points and actions as mapping features. The advantage of using physical actions for mapping is that they are ubiquitous across robotic platforms, even with varying sensing abilities. This is intended to allow heterogeneous robots to share navigation information and transfer knowledge between range and vision based SLAM techniques by providing a common means of understanding the environment.

Commonly, in existing SLAM techniques, if a series of landmarks (or images of the environment) are experienced again, then the robot reasons it has been in these positions previously, i.e. closing the loop. The insight in this work is that if a robot experiences a series of *actions* based on the environmental input regardless of the format of the input, then it can reason that it is in these positions on a learnt map.

CAM-SLAM has been trained and evaluated on real-world robotic platforms in an office environment. CAM-SLAM, through ACS and generalisation, was shown to autonomously generate CAMs capable of effectively and consistently navigating robots to given goal locations with 97% less mapping information than standard approaches.

The results have shown the successful sharing of CAMs between two heterogeneous robotic platforms with varying capabilities in a real-world environment. The ability to share mapping information between heterogeneous platforms is novel. This is useful for dangerous environments where a lesser capable robot (cheaper) can be used for the initial exploration before risking more capable (expensive) robot platforms.

Finally, the results have demonstrated the novel contribution of shar-

ing navigation information between two distinct SLAM approaches, range-based GMapping and vision-based RatSLAM. The advantage of sharing information between SLAM processes allows an individual robotic platform to utilise the best fit SLAM process for its sensors, while still providing useful navigation information for robots with different sensor types. This improves the communal knowledge of co-operative robotics solving given tasks.

7.2 Emotion inspired adaptive robotic path planning

Humans rely upon cognitive reasoning to determine their actions within the environment, however their reasoning ability is influenced by internal emotional states. Generally considered to be a detriment to a person's cognitive process, recent research is regarding emotions as a beneficial mechanism in the decision making process [23] [61]. Fellous [33] has suggested that biological emotions facilitate the transfer of simple, but high-impact information, both externally and internally, for operating system-like tasks. Inspired by these theories on natural emotions, this thesis has presented a novel emotion model designed to improve a robot's navigation performance through learning to adapt a rigid computational path-planning approach.

Emotions are considered to form the knots of bow-tie structures in biological cognitive systems, decoupling stimulus and response [65]. Inspired by this, the emotion model has been implemented based on the concept of a bow-tie structure by linking emotional reinforcers and behavioural modifiers through intermediary emotion states. An important function of emotions in the model is to provide a compact set of high-impact behaviour adaptations, reducing an otherwise tangled web of stimulus-response patterns into a compact and manageable structure. The key contribution is that the system is capable of learning the behavioural responses with no

human pre-specifying these responses. The emotion model co-operatively learns to associate emotional reinforcers to an evolving population of behavioural modifiers. The emotion system achieves this through learning novel Reinforcer-Emotion-Modifier classifiers through an XCS, relying on a GA to evolve the population of modifiers.

The emotion model was initially trained and evaluated in the simulated offices of Willow Garage. The results of this training demonstrated the emotion model is capable of learning up to three emotion states for robotic navigation. The three emotions that were judged to have emerged from the emotion model were labelled: *fear*, *apprehension*, and *happiness*. The fear and apprehension responses slow the robot's speed and drive the robot away from obstacles when the robot experiences pain (collisions with the environment). The happiness response increases the speed of the robot and reduces the safety margins around obstacles when pain is absent, allowing the robot to drive closer to obstacles in the environment.

However, the emotion model has been shown to be incapable of learning beneficial behaviours with four or more emotion states. An investigation into the fitness of the modifier and classifier populations indicates that the classifiers are reliant on the GA evolving beneficial behaviours (modifiers that achieve a high fitness). Increasing the number of potential emotions in the model reduces the overall fitness of the modifier population by increasing the search space and spreading the reward across more modifiers, reducing the effectiveness of the learned bow-ties. It can therefore be argued that in structured and static environments, such as an office, only three base emotions are needed to assist navigation.

It has been demonstrated that these learned emotion responses have improved the navigation performance of the robot by reducing collisions and navigation times, when compared with the default rigid navigation system. The two emotion model (fear and happiness) improved performance the most, indicating that a robot may only require two emotion states (fear and happiness) to navigate.

The emotion model has been verified in the real-world by training the emotion model in the office environment at VUW with two possible emotion states. The emerging emotion states had similar emotional responses to the fear and happiness responses from the simulated training. Compared with the rigid path-planning approach, the emotion model was able to reduce collisions and improve navigation time in the real-world environment. It was also demonstrated that the emotion model learned in simulation could be transferred to the real-world, by also showing an improved navigation performance in the real-world. This demonstrates that the emotion model can learn beneficial behaviours for robotic navigation, with both simulated and real-world models improving the navigation performance of the rigid navigation system.

7.3 Summary of contributions

This thesis has introduced a beneficial learning mechanism to the problems of localisation and path-planning. Both of the presented systems utilise a LCS to allow the robot to learn from its interaction with the environment. The LCS is ideal for robotics due to its ability to: generalise states to reduce memory and computation, handle sparse and unspecific reward through reinforcement learning, produce human readable state-action rules, and handle noisy real-world input. As has been demonstrated in this work, the LCS has been able to successfully reduce the required navigation information, learn anticipations for path-planning, and compact the stimulus-response state space while being trained with noisy robotic sensor data. However, a common problem when applying machine learning techniques to robotics is the time required for the robot to physically traverse the environment and the number of learning iterations required for the system to learn.

In the case of CAM-SLAM the ACS requires numerous iterations to learn both the decision points and the actions associating them, requiring

the robot to traverse the environment a number of times. Although sharing navigation information has demonstrated reduced learning times, the initial learning still requires a number of learning iterations before an effective CAM is created. In time critical conditions this can hinder a robots efforts to complete a task, e.g. effectively locate a person in relief efforts.

Similarly, the emotion model took a number of days to learn the emotional behaviours. The emotion model does have the benefit of being able to transfer knowledge learned in simulation to the real-world, but this still requires training the emotion model for novel environments and for different platforms.

Learning is what allows humans to be effective at solving problems, and is considered to be a function required for autonomous robotics to become a reality. However, a trade-off between required learning times and resulting capabilities needs to be considered when applying learning systems to real-world applications.

7.4 Future Work

The environments CAM-SLAM is capable of operating in are limited by its ability to match decision points based on sensor data. The simple sensor comparison approach used in this thesis is effective only for the specific domain the robots were trained in. To operate in complex domains, e.g. outdoors or long time duration experiments, the condition matching stage needs to be extended to incorporate pose and condition invariant approaches for matching sensor data.

The action set used in the training of CAM-SLAM was limited to four actions suitable for verifying CAM-SLAM in an office environment, but it is unrealistic in domains that contain a more continuous structure. The action set will need to be extended to incorporate a wider representation of possible actions. However, simply increasing the number of possible actions (physical angles the robot can move) will hinder the ability of the

ACS to learn the possible expectation mappings.

One potential solution would be to develop a set of actions contextualised to the specific location of a decision point. Reducing the arbitrary number of directions to a more manageable concept. For example, an action can refer to a door or opening at a given decision point, i.e. go through the 'door' or 'opening'. However, not all sensors are capable of extracting such complex features, a solution should be flexible for a range of sensor types. Another solution would be to use LCS with computed actions [44] or code fragmented actions [43], which enable continuous actions to be learnt provided they can be determined from environmental input.

As discussed in section 6.5, the problem of robotic navigation has no social interaction or community aspect, so it is not reasonable to consider that the emotion model can learn emotions such as guilt, pride, or love in the tested domains. Modern evolutionary theory proposes different emotions evolved at different times [40]. Primal emotions, such as fear, are associated with ancient parts of the brain and presumably evolved among our earlier ancestors. Affectionate emotions, such as love for an offspring, are considered to have evolved among early mammals. Social emotions, such as guilt and pride, evolved among social primates. Furthermore, more recently evolved parts of the brain can affect older parts of the brain, such as when the cortex moderates the amygdala's fear response.

The navigation task may only require primitive behaviours such as fear. This proposition is supported by the emotion model only producing 'fear' and 'happiness' responses. To learn more complex emotions within the presented emotion model the robot may need to learn in domains with social or co-operative tasks. For example, co-operative mapping or 'foraging' (searching) for a 'food' (power) supply.

Furthermore, the emotion model in its current implementation has no mechanism for behaviours to learn to affect 'older behaviours' or reinforcer inputs (such as pain influencing the sensation of touch), i.e. the current model has no feedback mechanism between the emotion and re-

inforcer inputs. To achieve this, the model could be extended to utilise XCSCFC (code fragmented LCS). Specifically, the code fragments can take previously learned rules and incorporate them into newer more complex rules, i.e. incorporate ‘older’ parts of the brain into ‘newer’ parts of the brain. The emotion model may then have the ability to learn emotions in increasingly complex environments, similar to the evolution of emotion in animals [40].

7.5 Publications

The following publications were produced during this thesis:

- WILLIAMS, H., BROWNE, W. N., AND MILFORD, M. Image region salience for improving appearance-based place recognition using a supervised classifier system. In *Proceedings of the 2012 Australasian Conference on Robotics & Automation* (2012), Australian Robotics & Automation Association
- WILLIAMS, H., BROWNE, W. N., AND CARNEGIE, D. A. Robotic competitions: short term pain for long term gain. In *Proceedings of the 2014 Australasian Conference on Robotics & Automation* (2014), Australian Robotics & Automation Association
- WILLIAMS, H., LEE-JOHNSON, C., BROWNE, W. N., AND CARNEGIE, D. A. Emotion inspired adaptive robotic path planning. In *Evolutionary Computation (CEC), 2015 IEEE Congress on* (2015), IEEE
- WILLIAMS, H., BROWNE, W. N., AND CARNEGIE, D. A. Learned action slam: Sharing slam through learned path planning information between heterogeneous robotic platforms. *Applied soft computing* (In review as of 2015)

- WILLIAMS, H., BROWNE, W. N., AND CARNEGIE, D. A. Human inspired heterogeneous robotic navigation. *Applied soft computing* (In process of writing)
- WILLIAMS, H., BROWNE, W. N., AND CARNEGIE, D. A. Emotion inspired adaptive robotic path planning. *Applied soft computing* (In process of writing)

7.6 Final Summary

This thesis has introduced two human inspired mechanisms to address limitations in current robotic navigation techniques. First, a cognitive mapping based technique named CAM-SLAM has been developed that utilises novel decision points as ubiquitous mapping features to facilitate autonomous sharing of navigation information between heterogeneous robotic platforms. Furthermore, this technique has been demonstrated as being capable of autonomously sharing navigation information between range-based and vision-based SLAM techniques for the first time. Second, an emotion inspired system has been developed to add flexibility to a rigid navigation system. The learning system has been shown to be capable of learning anthropomorphic emotional responses with fear, apprehension, and happiness emerging from training without human bias. It has been demonstrated that these learned emotion responses have improved the navigation performance of the robot by reducing collisions and navigation times.

Bibliography

- [1] ARKIN, R. Moving up the food chain: motivation and emotion in behavior-based robots. In *Who Needs Emotions? The Brain meets the robot*, J. Fellous, Ed. Oxford University Press, 2005, pp. 270–425. Cited on page 25.
- [2] BAILEY, T., NIETO, J., GUIVANT, J., STEVENS, M., AND NEBOT, E. Consistency of the ekf-slam algorithm. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on* (2006), IEEE, pp. 3562–3568. Cited on page 16.
- [3] BALL, D., HEATH, S., WILES, J., WYETH, G., CORKE, P., AND MILFORD, M. Openratslam: an open source brain-based slam system. *Autonomous Robots* 34, 3 (2013), 149–176. Cited on page 46.
- [4] BAY, H., TUYTELAARS, T., AND VAN GOOL, L. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*. Springer, 2006, pp. 404–417. Cited on page 17.
- [5] BELLMAN, R. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM* 4, 6 (1961), 284. Cited on page 32.
- [6] BONARINI, A. An introduction to learning fuzzy classifier systems. In *Learning Classifier Systems*. Springer, 2000, pp. 83–104. Cited on page 37.

- [7] BOULDING, K. E. *The image: Knowledge in life and society*, vol. 47. University of Michigan Press, 1961. Cited on page 6.
- [8] BREAZEAL, C. Emotion and sociable humanoid robots. *International Journal of Human-Computer Studies* 59, 1 (2003), 119–155. Cited on page 112.
- [9] BREAZEAL, C. Function meets style: insights from emotion theory applied to hri. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 34, 2 (2004), 187–194. Cited on pages 9, 25, and 136.
- [10] BROOKS, R., ET AL. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of* 2, 1 (1986), 14–23. Cited on page 113.
- [11] BULL, L. Learning classifier systems: A brief introduction. In *In Bull, L (Ed.): Applications of Learning Classifier Systems. Berlin u.a* (2004), Springer, p. 14. Cited on page 33.
- [12] BULL, L., AND KOVACS, T. Foundations of learning classifier systems: An introduction. In *Foundations of Learning Classifier Systems*. Springer, 2005, pp. 1–17. Cited on page 37.
- [13] BUTZ, A. M. V., GOLDBERG, B. D. E., AND STOLZMANN, C. W. The anticipatory classifier system and genetic generalization. *Natural Computing* 1, 4 (2002), 427–467. Cited on page 36.
- [14] BUTZ, M. V. Learning classifier systems. In *Springer Handbook of Computational Intelligence*. Springer, 2015, pp. 961–981. Cited on page 43.
- [15] BUTZ, M. V., KOVACS, T., LANZI, P. L., AND WILSON, S. W. How xcs evolves accurate classifiers. In *Proceedings of the Third Genetic and*

- Evolutionary Computation Conference (GECCO-2001)* (2001), Citeseer, pp. 927–934. Cited on page 40.
- [16] BUTZ, M. V., AND STOLZMANN, W. An algorithmic description of acs2. In *Advances in learning classifier systems*. Springer, 2002, pp. 211–229. Cited on page 43.
- [17] BUTZ, M. V., AND WILSON, S. W. An algorithmic description of xcs. In *Advances in Learning Classifier Systems*. Springer, 2001, pp. 253–272. Cited on page 38.
- [18] CHAMBON, V., WENKE, D., FLEMING, S. M., PRINZ, W., AND HAGGARD, P. An online neural substrate for sense of agency. *Cerebral Cortex* 23, 5 (2013), 1031–1037. Cited on page 106.
- [19] CRAIG, A. A new view of pain as a homeostatic emotion. *Trends in neurosciences* 26, 6 (2003), 303–307. Cited on page 151.
- [20] CSETE, M., AND DOYLE, J. Bow ties, metabolism and disease. *TRENDS in Biotechnology* 22, 9 (2004), 446–450. Cited on page 22.
- [21] CUMMINS, M., AND NEWMAN, P. Highly scalable appearance-only slam-fab-map 2.0. In *Robotics: Science and Systems* (2009), vol. 5, Seattle, USA. Cited on page 17.
- [22] DAVISON, A. J., REID, I. D., MOLTON, N. D., AND STASSE, O. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 29, 6 (2007), 1052–1067. Cited on page 17.
- [23] DAWKINS, M. S. Animal minds and animal emotions. *American Zoologist* 40, 6 (2000), 883–888. Cited on pages 8, 24, and 155.
- [24] DEARDEN, A., AND DEMIRIS, Y. Learning forward models for robots. In *IJCAI* (2005), vol. 5, p. 1440. Cited on page 29.

- [25] DILLMANN, R., ROGALLA, O., EHRENMANN, M., ZOLLNER, R., AND BORDEGONI, M. Learning robot behaviour and skills based on human demonstration and advice: the machine learning paradigm. In *ROBOTICS RESEARCH-INTERNATIONAL SYMPOSIUM-* (2000), vol. 9, pp. 229–238. Cited on page 29.
- [26] DISSANAYAKE, M., NEWMAN, P., CLARK, S., DURRANT-WHYTE, H. F., AND CSORBA, M. A solution to the simultaneous localization and map building (slam) problem. *Robotics and Automation, IEEE Transactions on* 17, 3 (2001), 229–241. Cited on pages 1 and 2.
- [27] DORIGO, M., AND SCHNEPF, U. Genetics-based machine learning and behavior-based robotics: a new synthesis. *Systems, Man and Cybernetics, IEEE Transactions on* 23, 1 (1993), 141–154. Cited on page 29.
- [28] DOUCET, A., DE FREITAS, N., MURPHY, K., AND RUSSELL, S. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence* (2000), Morgan Kaufmann Publishers Inc., pp. 176–183. Cited on page 46.
- [29] DOWNS, R. M., AND STEA, D. *Cognitive maps and spatial behavior: Process and products*. na, 1973. Cited on page 6.
- [30] EIBEN, A. E., AND SMITH, J. E. *Introduction to evolutionary computing*. Springer Science & Business Media, 2003. Cited on pages 30 and 34.
- [31] EKMAN, R. Facial expression and emotion. *American Psychologist* 48 (1993), 384–392. Cited on pages 24 and 28.
- [32] FEDER, H. J. S., LEONARD, J. J., AND SMITH, C. M. Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research* 18, 7 (1999), 650–668. Cited on page 18.

- [33] FELLOUS, J.-M. From human emotions to robot emotions. *Architectures for Modeling Emotion: Cross-Disciplinary Foundations*, American Association for Artificial Intelligence (2004), 39–46. Cited on pages 8, 24, 101, and 155.
- [34] FERGUSON, D., LIKHACHEV, M., AND STENTZ, A. A guide to heuristic-based path planning. In *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)* (2005), pp. 9–18. Cited on page 21.
- [35] FILLIAT, D., AND MEYER, J.-A. Map-based navigation in mobile robots:: I. a review of localization strategies. *Cognitive Systems Research* 4, 4 (2003), 243–282. Cited on page 19.
- [36] FISKE, A. P. *Structures of social life: The four elementary forms of human relations: Communal sharing, authority ranking, equality matching, market pricing*. Free Press, 1991. Cited on page 4.
- [37] FOGARTY, L., STRIMLING, P., AND LALAND, K. N. The evolution of teaching. *Evolution* 65, 10 (2011), 2760–2770. Cited on pages 4 and 153.
- [38] GOLDBERG, D. E., AND HOLLAND, J. H. Genetic algorithms and machine learning. *Machine learning* 3, 2 (1988), 95–99. Cited on pages 4, 34, and 40.
- [39] GUTMANN, J.-S., AND KONOLIGE, K. Incremental mapping of large cyclic environments. In *Computational Intelligence in Robotics and Automation, 1999. CIRA'99. Proceedings. 1999 IEEE International Symposium on* (1999), IEEE, pp. 318–325. Cited on page 90.
- [40] HESS, U., AND THIBAUT, P. Darwin and emotion expression. *American Psychologist* 64, 2 (2009), 120. Cited on pages 150, 159, and 160.

- [41] HOLLINGER, G., GEORGIEV, Y., MANFREDI, A., MAXWELL, B. A., PEZZEMENTI, Z., MITCHELL, B., ET AL. Design of a social mobile robot using emotion-based decision mechanisms. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on* (2006), IEEE, pp. 3093–3098. Cited on page 9.
- [42] HU, Y., AND YANG, S. X. A knowledge based genetic algorithm for path planning of a mobile robot. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on* (2004), vol. 5, IEEE, pp. 4350–4355. Cited on page 22.
- [43] IQBAL, M., BROWNE, W. N., AND ZHANG, M. Extracting and using building blocks of knowledge in learning classifier systems. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation* (2012), ACM, pp. 863–870. Cited on pages 36 and 159.
- [44] IQBAL, M., BROWNE, W. N., AND ZHANG, M. Xcsr with computed continuous action. In *AI 2012: Advances in Artificial Intelligence*. Springer, 2012, pp. 350–361. Cited on pages 36 and 159.
- [45] IQBAL, M., BROWNE, W. N., AND ZHANG, M. Reusing building blocks of extracted knowledge to solve complex, large-scale boolean problems. *Evolutionary Computation, IEEE Transactions on* 18, 4 (2014), 465–480. Cited on page 31.
- [46] ISEN, A. M. An influence of positive affect on decision making in complex situations: Theoretical issues with practical implications. *Journal of consumer psychology* 11, 2 (2001), 75–85. Cited on page 112.
- [47] ISEN, A. M. An influence of positive affect on decision making in complex situations: Theoretical issues with practical implications. *Journal of Consumer Psychology* 11, 2 (2001), 75–85. Cited on page 136.
- [48] ISHIBUCHI, H., NAKASHIMA, T., AND MURATA, T. Performance evaluation of fuzzy classifier systems for multidimensional pattern

- classification problems. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 29, 5 (1999), 601–618. Cited on page 37.
- [49] ISHII, H., SHI, Q., FUMINO, S., KONNO, S., KINOSHITA, S., OKABAYASHI, S., IIDA, N., KIMURA, H., TAHARA, Y., SHIBATA, S., ET AL. A novel method to develop an animal model of depression using a small mobile robot. *Advanced Robotics* 27, 1 (2013), 61–69. Cited on pages 28 and 103.
- [50] JACOBS, L. F., AND SCHENK, F. Unpacking the cognitive map: the parallel map theory of hippocampal function. *Psychological review* 110, 2 (2003), 285. Cited on page 6.
- [51] JACOBSON, A., AND MILFORD, M. Towards brain-based sensor fusion for navigating robots. In *Proceedings of the 2012 Australasian Conference on Robotics & Automation* (2012), Australian Robotics & Automation Association. Cited on pages 5 and 49.
- [52] KAI, S., QI, W., AND MINGLI, D. Approach to nonlinear blind source separation based on niche genetic algorithm. In *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on* (2006), vol. 1, IEEE, pp. 441–445. Cited on page 22.
- [53] KARUMANCHI, S., ALLEN, T., BAILEY, T., AND SCHEDING, S. Non-parametric learning to aid path planning over slopes. *The International Journal of Robotics Research* 29, 8 (2010), 997–1018. Cited on page 22.
- [54] KOBER, J., BAGNELL, J. A., AND PETERS, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* (2013), 0278364913495721. Cited on pages 30, 31, and 33.
- [55] KONOLIGE, K., AND AGRAWAL, M. Frameslam: From bundle adjustment to real-time visual mapping. *Robotics, IEEE Transactions on* 24, 5 (2008), 1066–1077. Cited on page 17.

- [56] LATOMBE, J.-C. *Robot motion planning*, vol. 124. Springer Science & Business Media, 2012. Cited on page 22.
- [57] LAVALLE, S. M. *Planning algorithms*. Cambridge university press, 2006. Cited on page 18.
- [58] LEE-JOHNSON, C. P., AND CARNEGIE, P. D. A. *Robotic Emotions: Navigation with Feeling*. Information Science Reference, 2009, pp. 88 – 117. Cited on pages 9, 25, 26, 29, 48, and 103.
- [59] LEI, L., WANG, H., AND WU, Q. Improved genetic algorithms based path planning of mobile robot under dynamic unknown environment. In *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on* (2006), IEEE, pp. 1728–1732. Cited on page 22.
- [60] LEONARD, J. J., AND DURRANT-WHYTE, H. F. Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems' 91. Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on* (1991), Ieee, pp. 1442–1447. Cited on pages 14 and 16.
- [61] LEVENTHAL, H., AND SCHERER, K. The relationship of emotion to cognition: A functional approach to a semantic controversy. *Cognition and emotion* 1, 1 (1987), 3–28. Cited on pages 8, 24, and 155.
- [62] LI, Y., AND OLSON, E. B. A general purpose feature extractor for light detection and ranging data. *Sensors* 10, 11 (2010), 10356–10375. Cited on page 51.
- [63] LOWE, D. G. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on* (1999), vol. 2, Ieee, pp. 1150–1157. Cited on page 17.

- [64] LOWRY, S. M., MILFORD, M. J., AND WYETH, G. F. Transforming morning to afternoon using linear regression techniques. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on* (2014), IEEE, pp. 3950–3955. Cited on page 17.
- [65] MACLENNAN, B. J. Robots react, but can they feel? *A Protophenomenological Analysis*. In *Handbook of Research on Synthetic Emotions and Sociable Robotics: New Applications in Affective Computing and Artificial Intelligence*, edited by Jordi Vallverdú and David Casacuberta (2009), 133–53. Cited on pages 9, 102, and 155.
- [66] MADDERN, W., MILFORD, M., AND WYETH, G. Cat-slam: probabilistic localisation and mapping using a continuous appearance-based trajectory. *The International Journal of Robotics Research* 31, 4 (2012), 429–451. Cited on page 17.
- [67] MARDER-EPPSTEIN, E., BERGER, E., FOOTE, T., GERKEY, B., AND KONOLIGE, K. The office marathon: Robust navigation in an indoor office environment. In *International Conference on Robotics and Automation* (2010). Cited on page 46.
- [68] MCMANUS, C., UPCROFT, B., AND NEWMAN, P. Scene signatures: Localised and point-less features for localisation. In *Proceedings of Robotics Science and Systems (RSS)* (Berkeley, CA, USA, July 2014). Cited on page 17.
- [69] MEYER, J.-A., AND FILLIAT, D. Map-based navigation in mobile robots:: Ii. a review of map-learning and path-planning strategies. *Cognitive Systems Research* 4, 4 (2003), 283–317. Cited on page 19.
- [70] MILFORD, M. J., AND WYETH, G. F. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on* (2012), IEEE, pp. 1643–1649. Cited on pages 18 and 59.

- [71] MILFORD, M. J., WYETH, G. F., AND RASSER, D. Ratslam: a hippocampal model for simultaneous localization and mapping. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on* (2004), vol. 1, IEEE, pp. 403–408. Cited on page 17.
- [72] MILLER, G. A. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review* 63, 2 (1956), 81. Cited on page 28.
- [73] MONTEMERLO, M., THRUN, S., KOLLER, D., AND WEGBREIT, B. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)* (Acapulco, Mexico, 2003), IJCAI. Cited on page 16.
- [74] MONTEMERLO, M., THRUN, S., KOLLER, D., WEGBREIT, B., ET AL. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI* (2002), pp. 593–598. Cited on page 16.
- [75] MOSHKINA, L., AND ARKIN, R. Human perspective on affective robotic behavior: a longitudinal study. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on* (Aug 2005), pp. 1444–1451. Cited on page 25.
- [76] NUSKE, S., ROBERTS, J., AND WYETH, G. Robust outdoor visual localization using a three-dimensional-edge map. *Journal of Field Robotics* 26, 9 (2009), 728–756. Cited on page 17.
- [77] OTTE, M. W. A survey of machine learning approaches to robotic path-planning. Cited on pages 3 and 22.
- [78] PAWLICZEK, C. M., DERNTL, B., KELLERMANN, T., GUR, R. C., SCHNEIDER, F., AND HABEL, U. Anger under control: neural cor-

- relates of frustration as a function of trait aggression. *PloS one* 8, 10 (2013), e78503. Cited on page 112.
- [79] PENG, J. X., THOMPSON, S., AND LI, K. A gradient-guided niche method in genetic algorithm for solving continuous optimization problems. In *Proceedings of the fourth World Congress on Intelligent Control and Automation June* (2002), pp. 10–14. Cited on page 22.
- [80] PEPPERELL, E., CORKE, P., AND MILFORD, M. Towards vision-based pose-and condition-invariant place recognition along routes. In *Proceedings of the Australasian Conference on Robotics and Automation 2014* (2014), Australian Robotics & Automation Association ARAA. Cited on page 18.
- [81] PEPPERELL, E., CORKE, P., MILFORD, M. J., ET AL. All-environment visual place recognition with smart. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on* (2014), IEEE, pp. 1612–1618. Cited on page 18.
- [82] PIPE, A. G., AND CARSE, B. Fuzzy classifier system architectures for mobile robotics: An experimental comparison. *International Journal of Intelligent Systems* 22, 9 (2007), 993–1019. Cited on page 37.
- [83] PLUTCHIK, R. *A general psychoevolutionary theory of emotion*. Academic press, New York, 1980, pp. 3–33. Cited on pages 24 and 111.
- [84] POWELL, W. B. Ai, or and control theory: A rosetta stone for stochastic optimization. Tech. rep., Technical report, Princeton University, 2012. Cited on page 31.
- [85] QUIGLEY, M., CONLEY, K., GERKEY, B., FAUST, J., FOOTE, T., LEIBS, J., WHEELER, R., AND NG, A. Y. Ros: an open-source robot operating system. In *ICRA workshop on open source software* (2009), no. 3.2, p. 5. Cited on page 45.

- [86] ROLLS, E. Emotion explained. In *Who Needs Emotions? The Brain meets the robot*, J. Fellous, Ed. Oxford University Press, 2005, pp. 117–146. Cited on pages 26, 27, 28, 105, 106, and 112.
- [87] ROLLS, E. What are emotions, why do we have emotions, and what is their computational basis? In *Who Needs Emotions? The Brain meets the robot*, J. Fellous, Ed. Oxford University Press, 2005, pp. 117–146. Cited on pages 26, 103, 105, 106, and 112.
- [88] ROLLS, E. T. Precis of the brain and emotion. *Behav. Brain Sci* 23, 2 (2000), 177–191. Cited on page 24.
- [89] SIEGWART, R., NOURBAKHSI, I. R., AND SCARAMUZZA, D. *Introduction to autonomous mobile robots*. MIT press, 2011. Cited on page 2.
- [90] SINGH, S. P., BARTO, A. G., GRUPEN, R., AND CONNOLLY, C. Robust reinforcement learning in motion planning. *Advances in neural information processing systems* (1994), 655–655. Cited on page 22.
- [91] SMITH, P. J., MCCOY, C. E., AND LAYTON, C. Brittleness in the design of cooperative problem-solving systems: The effects on user performance. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 27, 3 (1997), 360–371. Cited on page 3.
- [92] SMITH, R. C., AND CHEESEMAN, P. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research* 5, 4 (1986), 56–68. Cited on page 14.
- [93] STENTZ, A., ET AL. The focussed d* algorithm for real-time replanning. In *IJCAI* (1995), vol. 95, pp. 1652–1659. Cited on page 21.
- [94] STOLZMANN, W. An introduction to anticipatory classifier systems. In *Learning Classifier Systems*. Springer, 2000, pp. 175–194. Cited on pages 36 and 44.

- [95] SÜNDERHAUF, N., NEUBERT, P., AND PROTZEL, P. Predicting the change—a step towards life-long operation in everyday environments. *Robotics Challenges and Vision (RCV2013)* (2014). Cited on page 17.
- [96] SUTTON, R. S., BARTO, A. G., AND WILLIAMS, R. J. Reinforcement learning is direct adaptive optimal control. *Control Systems, IEEE* 12, 2 (1992), 19–22. Cited on page 31.
- [97] TOMKINS, S. *Affect Imagery Consciousness: Volume II: The Negative Affects*. Springer Publishing Company, 1963. Cited on pages 113 and 136.
- [98] VAN WILGEN, C. P., AND KEIZER, D. The sensitization model to explain how chronic pain exists without tissue damage. *Pain Management Nursing* 13, 1 (2012), 60–65. Cited on page 105.
- [99] VELÁSQUEZ, J. D., AND MAES, P. Cathexis: A computational model of emotions. In *Proceedings of the First International Conference on Autonomous Agents* (New York, NY, USA, 1997), AGENTS '97, ACM, pp. 518–519. Cited on page 25.
- [100] VENTURES, M. D. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics* 23, 9 (2006), 661–692. Cited on page 16.
- [101] WARWICK, K., AND NASUTO, S. J. Historical and current machine intelligence. *Instrumentation & Measurement Magazine, IEEE* 9, 6 (2006), 20–26. Cited on page 4.
- [102] WATKINS, C. J., AND DAYAN, P. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292. Cited on pages 34 and 42.
- [103] WILLIAMS, H., AND BROWNE, W. N. Integration of learning classifier systems with simultaneous localisation and mapping for au-

- tonomous robotics. In *Evolutionary Computation (CEC), 2012 IEEE Congress on* (2012), IEEE, pp. 1–8. Cited on page 74.
- [104] WILLIAMS, H., BROWNE, W. N., AND CARNEGIE, D. A. Robotic competitions: short term pain for long term gain. In *Proceedings of the 2014 Australasian Conference on Robotics & Automation* (2014), Australian Robotics & Automation Association. Not cited
- [105] WILLIAMS, H., BROWNE, W. N., AND CARNEGIE, D. A. Emotion inspired adaptive robotic path planning. *Applied soft computing* (In process of writing). Not cited
- [106] WILLIAMS, H., BROWNE, W. N., AND CARNEGIE, D. A. Human inspired heterogeneous robotic navigation. *Applied soft computing* (In process of writing). Not cited
- [107] WILLIAMS, H., BROWNE, W. N., AND CARNEGIE, D. A. Learned action slam: Sharing slam through learned path planning information between heterogeneous robotic platforms. *Applied soft computing* (In review as of 2015). Not cited
- [108] WILLIAMS, H., BROWNE, W. N., AND MILFORD, M. Image region salience for improving appearance-based place recognition using a supervised classifier system. In *Proceedings of the 2012 Australasian Conference on Robotics & Automation* (2012), Australian Robotics & Automation Association. Not cited
- [109] WILLIAMS, H., LEE-JOHNSON, C., BROWNE, W. N., AND CARNEGIE, D. A. Emotion inspired adaptive robotic path planning. In *Evolutionary Computation (CEC), 2015 IEEE Congress on* (2015), IEEE. Not cited
- [110] WILSON, S. W. Zcs: A zeroth level classifier system. *Evolutionary computation* 2, 1 (1994), 1–18. Cited on page 34.

- [111] WILSON, S. W. Classifier fitness based on accuracy. *Evolutionary computation* 3, 2 (1995), 149–175. Cited on pages 34 and 36.
- [112] WILSON, S. W., AND GOLDBERG, D. E. A critical review of classifier systems. In *Proceedings of the third international conference on Genetic algorithms* (1989), Morgan Kaufmann Publishers Inc., pp. 244–255. Cited on page 34.
- [113] WOLPERT, D. M., GHAHRAMANI, Z., AND FLANAGAN, J. R. Perspectives and problems in motor learning. *Trends in cognitive sciences* 5, 11 (2001), 487–494. Cited on page 30.
- [114] WOOD, J. V., SALTZBERG, J. A., AND GOLDSAMT, L. A. Does affect induce self-focused attention? *Journal of personality and social psychology* 58, 5 (1990), 899. Cited on page 103.
- [115] WOŹNIAK, M., GRAÑA, M., AND CORCHADO, E. A survey of multiple classifier systems as hybrid systems. *Information Fusion* 16 (2014), 3–17. Cited on page 37.
- [116] ZANG, Z., LI, D., AND WANG, J. Learning classifier systems with memory condition to solve non-markov problems. *Soft Computing* 19, 6 (2012), 1679–1699. Cited on page 36.
- [117] ZATUCHNA, Z. V. Agentp model: Learning classifier system with associative perception. In *Parallel Problem Solving from Nature-PPSN VIII* (2004), Springer, pp. 1172–1181. Cited on page 37.
- [118] ZATUCHNA, Z. V., AND BAGNALL, A. J. A learning classifier system for mazes with aliasing clones. *Natural Computing* 8, 1 (2009), 57–99. Cited on pages 36 and 42.
- [119] ZHANG, A. M., AND KLEEMAN, L. Robust appearance based visual route following for navigation in large-scale outdoor environments. *The International Journal of Robotics Research* 28, 3 (2009), 331–356. Cited on page 59.