Emotion-based Parameter Modulation for a Mobile Robot Planning and Control System

A thesis

submitted in fulfilment

of the requirements for the Degree

of

Doctor of Philosophy in

Electronic and Computer System Engineering

at

Victoria University of Wellington

by

Christopher Peter Lee-Johnson

2008

Abstract

The hypothesis that artificial emotion-like mechanisms can improve the adaptive performance of robots and intelligent systems has gained considerable support in recent years. While artificial emotions are typically employed to facilitate human-machine interaction, this thesis instead focuses on modelling emotions and affect in a non-social context. In particular, affective mechanisms are applied to the problem of mobile robot navigation.

A three-layered reactive/deliberative controller is developed and implemented, resulting in several contributions to the field of mobile robot control. Rather than employing a reactive layer, a deliberative layer and an interface between them, the control problem is decomposed into three different conceptual spaces – position space, direction space and velocity space – with a distinct control layer applied to each. Existing directional and velocity space approaches such as the vector field histogram (VFH) and dynamic window methods employ different underlying mechanisms and terminology. This thesis unifies these approaches in order to compare and combine them. The weighted sum objective functions employed by some existing approaches that inspired the presented directional and velocity control layers are replaced by weighted products. This enables some hard constraints to be relaxed in favour of weighted contributions, potentially improving a system's flexibility without sacrificing safety (but coming at a cost to efficiency).

An affect model is developed that conceptualises emotions and other affective interactions as modulations of cognitive processes. Unlike other models of affect-modulated cognition (e.g. Dörner and Hille, 1995), this model is designed specifically to address problems relating to mobile robot navigation. The role of affect in this model is to continuously adapt a controller's behaviour patterns in response to different environments and momentary conditions encountered by the robot. Affective constructs such as moods and emotions are represented as intensity values that arise from hard-coded interpretations of local stimuli, as well as from learned associations stored in global maps. They are expressed as modulations of control parameters and location-specific biases to path-planning. Extensive simulation experiments are conducted in procedurally-generated environments to assess the performance contributions of this model and its individual components.

Acknowledgements

I wish to express my gratitude to the numerous people who have helped me during this project. First, I would like to thank my supervisor Professor Dale Carnegie for his invaluable advice and encouragement. I am also tremendously grateful for the time and effort he put into reviewing the numerous papers and thesis chapters I threw at him.

Also deserving of thanks are my fellow graduate students, past and present, who have exchanged ideas and provided assistance and support. In particular, I would like to thank Praneel Chand for providing feedback, contributing ideas and collaborating on a conference publication, and Johnny McClymont for his work on MARVIN's hardware. Other students who have assisted me are Jason Edwards, Thomas Roehr, David Williamson, Luke Cawley and Ashil Prakash. I also appreciate the technical and administrative assistance provided by Scott Forbes and Bruce Rhodes.

Finally I would like to acknowledge the New Zealand Tertiary Education Commission and Victoria University of Wellington for funding this research.

Table of Contents

List of Figures

List of Tables

1 Introduction

Despite decades of optimism, the robotics and artificial intelligence communities have thus far been unable to synthesize adaptive capabilities comparable to those possessed by humans or even simple animals. Futurists have predicted that strong AI (machine consciousness[1]) is "several decades away" since the 1950s, but over fifty years later it remains firmly in the realm of science fiction. Due to the difficulties associated with subjective issues such as consciousness, most current research (including this thesis) focuses on weak AI applications. Such research typically investigates computational mechanisms inspired by psychological and/or neurobiological models, but stops short of claiming that these software constructs are conscious in the same manner as humans or animals.

The problem of understanding the human mind has been similarly elusive. Fundamental philosophical issues such as the mind-body problem have been subject to endless speculation, but remain unsolved. Some critical elements may be missing from our picture of the mind. One of these elements may be emotion. Works of fiction often depict future artificial intelligences as emotionless beings driven by logic. However, as the classical distinction between emotion and cognition becomes increasingly nebulous, some authors have hypothesised that emotion may be a prerequisite for general intelligence (Minsky, 1986; Damasio, 1994). This view is best summarised by Minsky (1986):

> "The question is not whether intelligent machines can have any emotions, but whether machines can be intelligent without emotions."

Recent years have seen increased interest in the development of robots and intelligent systems that possess emotion-inspired software mechanisms. Researchers typically focus on the application of emotions to social robotics (e.g. Breazeal, 2004; Broekens, 2007; Hollinger et al., 2006). In this domain, the natural human tendency towards anthropomorphism can be exploited by portraying robotic emotions as facial expressions, body language and/or tone of voice. Emotions can also be applied to robotics applications other than human-machine interaction (Arkin, 2005). Nevertheless, purportedly general-purpose emotion models are often applied in a

[1] An in-depth discussion of machine consciousness is given in (Haikonen, 2003).

social context, where they influence interactions with humans or other robots. Few implementations have been demonstrated that approach the issue from the perspective of an individual robot in a non-social context.

In the biological world, emotion is often viewed as a general facilitator of survival and adaptation. Some emotional functions that are beneficial to humans and animals may also be applicable to the control of autonomous mobile robots. Like biological organisms, mobile robots interact with unpredictable, partially-known, real-world environments. They must contend with continuous, noisy, imprecise sensor data, rather than discrete labelled and categorised objects. Operating in real-time with limited resources, they must make tradeoffs between many competing factors (e.g. the need for fast action and careful planning; immediate reward and delayed gratification; efficiency and optimality; safety and success). Existing control methods have addressed many of these issues, but the general problems associated with mobile robot control remain non-trivial.

A number of control approaches have been developed that allow mobile robots to navigate safely and efficiently in various different types of environments. These include behaviour-based control methods utilising simple state machines; biologically-inspired systems based on artificial neural networks, fuzzy inference systems and/or evolutionary algorithms; and search-based motion planners that employ detailed world representations. Many successful controllers incorporate multiple techniques (e.g. reactive control and deliberative planning), combining their strengths and mitigating individual weaknesses. Traditional controllers are typically tuned, trained or evolved in a specific type of environment or a set of similar environments. At best, a robot is likely to perform sub-optimally if it is placed in an environment that differs markedly from the ones in which its controller is configured (e.g. if it is moved from a cluttered indoor laboratory to a sparsely occupied outdoor setting); at worst, it may fail utterly. Rigidity can be lessened by 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.

To address problems related to adaptive control and learning, several mobile robot controllers incorporating emotion-like software mechanisms have been developed (e.g. Gadanho and Hallam, 2001; Neal and Timmis, 2003). These systems typically

employ simple behaviour-based or neural network control architectures that are driven by emotions. However, quantitative results are scarce, and it is difficult to envision how such simple emotion-based control mechanisms could outperform traditional approaches that have proven highly reliable, albeit inflexible. Reliability will likely be compromised if a robot's decisions and actions are completely dominated by emotions, causing failures that an 'emotionless' robot can easily avoid.

Performance may be improved by combining traditional control approaches with emotion-like computational models that facilitate adaptation. In particular, 'artificial emotions' could be employed to modify a robot's behaviour in response to different environments and situations that it encounters as it performs its duties. While emotion should have a tangible influence over the robot's actions, in many situations it should remain a secondary influence rather than a dominant one. The implementation of such complementary systems could not only lead to the improvement of future mobile robot controllers, but it may inspire further advances in the ongoing multidisciplinary effort to understand and define biological cognition and emotion.

1.1 Objectives

The broad objective of this thesis is to address the question of whether computational mechanisms inspired by biological emotions can be beneficial to robots beyond the social domain. In particular, the thesis will focus on the emotion-based control of a mobile robot whose task is to navigate and explore arbitrary flat-surfaced environments. Different theories regarding biological and computational emotion will be investigated, but an emphasis will be placed on ideas that can be applied to our intended application. Given the subjective nature of biological emotions, no claims will be made that the software mechanisms investigated are 'real' emotions experienced by robots in the same manner as humans perceive emotions. The absence of such claims does not preclude the usefulness of this approach, however. A loose analogy can be made between artificial emotions and artificial neural networks (ANNs). ANNs are widely regarded as effective computational mechanisms even though few researchers would claim that they bear more than a superficial resemblance to their biological analogues. Similarly, this thesis will endeavour to determine whether emotion-like states and processes can be useful for the design of

robots and intelligent systems, even if they are not the same as their biological namesakes.

A robot controller comprising two main components will be developed and implemented. The first component is a cognitive system, represented by a control architecture that provides the robot's underlying navigation capabilities. As the robot is expected to respond to environmental dynamics while navigating in a complex world, this architecture will combine reactive control and deliberative planning approaches. While robustness is a high priority, it is also important for this system to be highly flexible, eschewing binary decisions and hard limits in favour of continuous weights and other parameters that can be adjusted to suit a wide variety of environments and situations.

The second component is an emotion system that influences the controller's behavioural tendencies in response to different emotion-like states and processes. This system will be inspired by biological theories of human emotion. However, the 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. Where biological plausibility comes at a cost to practical utility, this model will generally favour the latter.

To assess the model's effects on robot behaviour, the planning and control architecture's performance while operating in isolation will be compared to its performance when it is influenced by emotions. Experiments will be conducted in a diverse set of simulated environments with different cognitive and emotional components set to varying degrees of activation so that their individual performance contributions can be quantitatively measured.

1.2 Thesis Outline

- Chapter 2 – The question of what it means to model emotions in software agents is addressed, and the concept of affect (encompassing many emotion-like states and processes) is introduced. Psychological and neurobiological perspectives on biological emotions are described. Various different approaches for modelling emotions and related states and processes in intelligent systems and robots are reviewed.

- Chapter 3 – A high level description of the model of robotic affect developed for this project is presented. This model interacts with an underlying planning and control system by modulating its control parameters. It incorporates multiple different affective states, including stimuli, drives, emotions and moods as distinct, interacting components.
- Chapter 4 – MARVIN, the mobile robot that serves as the target platform and inspiration for this research, is introduced. The robot's hardware and software are described. A simulated version of MARVIN is developed for the purpose of testing our approach. Simulation maps can be generated procedurally, resulting in diverse sets of environments in which to conduct navigation experiments.
- Chapter 5 – The underlying planning and control system is implemented. A hybrid reactive/deliberative architecture is employed that incorporates three distinct control layers. The deliberative layer updates a set of rectangular grid maps and plans optimal paths through the global environment. Next, the directional control layer obtains a locally-optimal heading that loosely follows the planned path and avoids nearby obstacles. Finally, the velocity control layer obtains linear and angular velocities that move the robot in the intended direction at an appropriate speed while avoiding obstacles.
- Chapter 6 – Experimental results demonstrating the utility of the planning and control system's main components are presented. By varying certain weights, the performance contributions of individual subsystems can be measured.
- Chapter 7 – Navigation experiments are conducted on the integrated controller (without emotional influences). The performance effects of different control parameters are assessed so that they can be grouped into different categories and modulated appropriately by the affect model.
- Chapter 8 – Implementation details of the robotic affect model are provided. Affective stimuli grow or decay in response to specific eliciting events. Drives represent groupings of control parameters that are modulated in a similar manner. Emotions bridge the gap between stimuli and drives, and also produce location-specific biases to path planning. Moods modulate the overall influence certain emotions can exert over the robot's behaviour.

- Chapter 9 – Experimental results showing the performance influences of different components of the affect model are presented. ‘Emotional’ navigation is compared to ‘emotionless’ navigation by iteratively varying weights controlling the contributions of different affective components.
- Chapter 10 – Results obtained from the fully-integrated affective navigation system are shown and discussed. Interactions between different components are demonstrated, and a set of experiments are conducted to quantitatively assess its performance in different types of environments. Some experiments involving extensions to this research (robotic personality and human reactions to affective mobile robots) are also conducted.
- Chapter 11 – The thesis concludes with a summary of the contributions and international publications resulting from this research, and a discussion of future work.

2 Robotic Emotions and Affective Computing

Emotions have historically been regarded as the antithesis of reason. This belief largely stems from the Cartesian dualism philosophies that have dominated Western thinking for centuries. Such philosophies tend to view emotions in a negative light, regarding them as base instincts that the rational mind should strive to overcome. Viewed from this perspective, it would appear counterproductive to attempt to bestow machines with emotions.

Recent years have seen growing acceptance of philosophies that challenge the strong division between mind and body. Increasing psychological and neurobiological evidence links emotions to functions that were once considered purely cognitive, such as problem-solving, learning, memory and perception. This has led some authors to conclude that emotions are a prerequisite for intelligence (Damasio, 1999; Minsky, 1986). A more moderate position adopted by other researchers is that emotions may not be necessary for intelligence, but they can be beneficial to adaptive behaviour (Picard, 2002; Sloman, 2005). Hence, the idea of applying emotion-like software mechanisms to machines deserves serious consideration.

2.1 Defining Robotic Emotions

There are no universally-accepted definitions for emotions. As with other subjective concepts such as consciousness, emotions are very difficult to define in concrete terms. Other emotion-like words such as feelings, moods, attitudes, drives and personality traits are similarly nebulous. To some authors, these terms are synonymous with emotions, while others regard them as distinct concepts. According to authors such as Sloman et al. (2005), it is more useful to regard emotion as a subset of the more general category of affect, which includes a wide range of related states and processes. Picard (1995) coined the term ‘affective computing’ to describe the application of emotions and affect to software systems.

The conceptual ambiguity is compounded when authors attempt to define emotions in the context of software agents and robots. At one end of the spectrum of definitions are those that regard emotions as states experienced subjectively by a machine in a manner akin to the way humans feel emotions. The problem with such strong

definitions is that science is ill-equipped to contend with subjective issues, particularly in nonhumans. This problem is similar to that encountered by researchers attempting to study animal emotions (Dawkins, 2000). Lacking complex language abilities, most animals cannot easily report their subjective feelings and experiences, so researchers can only study their behaviours. Certain animal behaviours outwardly resemble human emotions. However, to claim that they actually are emotions is to assume anthropomorphism. Hence, it is difficult to prove that animals experience emotions in the same way as humans, even though it is highly likely that some emotions do cross species boundaries. The case for subjectively experienced emotions in software agents or robots is even less convincing, because there are fewer similarities between humans and computers than there are between humans and animals.

The implementation of subjective emotions is dependent on (and perhaps required for) strong AI, machines capable of experiencing the subjective aspect of consciousness (Haikonen, 2003). Depending on an author's philosophical background, the viability of this type of implementation ranges from 'impossible' (some Western philosophies) to 'already achieved' (some Eastern philosophies). Regardless, any claims that existing robots or software agents can subjectively experience emotions are at present not particularly useful from an engineering perspective.

A more practical approach is to disregard the subjective awareness or representation of emotions within the machine, and focus on the portrayal of emotions in a manner that is believable to human observers. Thus, emotions are represented within the observer, rather than the machine portraying them. By this definition, even inanimate objects such as stuffed animals can be said to possess emotions if they are convincingly presented. This is the position adopted by many researchers who develop robots that interact 'emotionally' with humans. From a practical perspective, this type of implementation is the easiest to achieve, because it does not require an in-depth understanding of biological emotions.

This thesis adopts a moderate position between these two approaches. In the context of this research, artificial emotions are not 'real' emotions experienced subjectively by software agents or robots. Nor are they superficial external responses intended only to mimic human emotions. Rather, artificial emotions are software mechanisms inspired by theories of biological emotions that enable a machine to adapt to certain

situations that arise as it performs its duties. While this definition can apply to models that facilitate human-machine interaction, this thesis focuses largely on the effects of emotions on general performance.

2.2 Biological Emotions

Architectures for autonomous agents and robots currently bear little resemblance to the cognitive architectures of biological brains. Thus, until the complexity and generalised capabilities of robots and agents improve, there is little incentive to attempt to model the full range of biological emotions, or for artificial emotions to have as broad an influence over cognitive processes as their real-world counterparts (Sloman et al., 2005). Nevertheless, some properties of biological emotions may be useful for solving general computational problems encountered by the current generation of intelligent systems.

2.2.1 Psychology Perspective

Psychological studies of emotion generally consist of surveys of subjective experiences, or observations of human or animal behaviours, either in controlled experiments or in their natural habitats. Fundamental questions about biological emotions that have been studied include:

- *Categorisation* – Which emotions (if any) are neurologically distinct and universal among human cultures?
- *Elicitation* – Is the elicitation of emotions an innate response, a learned process or a combination of the two? To what extent does cognition contribute to emotional processes?
- *Functions* – What are the functional roles of emotions? To what extent do emotions influence cognitive processes such as memory and learning?

2.2.1.1 Categorisation

There are two main approaches to the categorisation of emotions. The first approach is to define a set of primary or basic emotions, each with distinct inputs and outputs, based on the theory that some emotions are largely innate, evolutionary in origin and

culturally universal. Ekman's cross-cultural research on human facial expressions supports this theory (Ekman, 1993). Results from surveys of many cultures worldwide, including remote villages in Papua New Guinea who had been largely isolated from the outside world, indicate that humans universally recognise facial expressions of anger, fear, happiness, sadness, surprise and disgust. Further evidence comes from studies of congenitally blind children which show that even humans who are unable to learn by observation of others can exhibit emotional facial expressions (Galati et al., 2001). According to Plutchik (2001), emotions should cross species as well as cultural boundaries to be defined as basic. Eight different primary emotions are grouped into pairs of polar opposites in Plutchik's (2001) model: anger/fear, happiness/sadness, anticipation/surprise and acceptance/disgust.

Other authors have proposed different lists of basic emotions, and different criteria by which they may be defined as basic. Table 2.1 shows some of the disparate lists collated by Ortony and Turner (1990). Many categorisations include equivalents of fear, anger, happiness and/or sadness (although they are often assigned different names), but there is little consensus on the existence and nature of other basic emotions.

TABLE 2.1: BASIC EMOTION LISTS

Reference	**Basic emotions**
Ekman, Friesen and Ellsworth (1982)	Anger, disgust, fear, joy, sadness, surprise
Gray (1982)	Rage and terror, anxiety, joy
Izard (1971)	Anger, contempt, disgust, distress, fear, guilt, interest, joy, shame, surprise
Oatley and Johnson-Laird (1987)	Anger, disgust, anxiety, happiness, sadness
Panksepp (1982)	Expectancy, fear, rage, panic
Plutchik (1980)	Acceptance, anger, anticipation, disgust, joy, fear, sadness, surprise
Tomkins (1984)	Anger, interest, contempt, disgust, shame, surprise
Weiner and Graham (1984)	Happiness, sadness

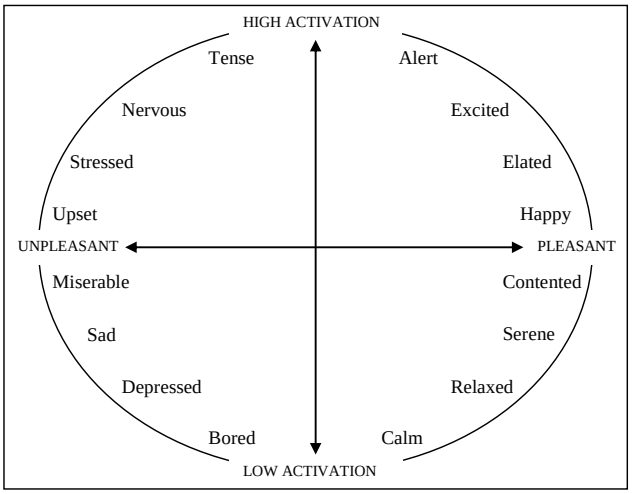

Figure 2.1: Russell's circumplex model represents emotions as locations on a 2D graph. The two dimensions modelled are pleasantness and activation.

The second approach is to represent emotions as regions of a continuous n-dimensional graph, where each dimension consists of a parameter such as valence (pleasantness) or arousal (intensity). One example is Russell's circumplex model (Russell, 1980), shown in Figure 2.1. This type of representation generally favours the theory that emotions are fluid, learned, culturally relative phenomena. Supporting evidence for this theory includes a study by Carroll and Russell (1996) which indicates that human interpretation of facial expressions can be influenced by the context in which they are presented. Wierzbicka (1992) argues that discrete emotions are cultural artefacts, explaining that words for certain emotions do not exist in some cultures (for example, the Ilongot and Ifaluk languages do not have a word describing anger).

2.2.1.2 Elicitation

Another source of contention among emotion researchers is the involvement of learning and cognition in the elicitation of emotions. At one end of the spectrum of theoretical approaches are somatic theories (e.g. the James-Lange theory), which regard emotions as the subjective experience of bodily states. For example, the sight of a lion might cause the body to freeze, increase heart rate and dilate the pupils. The brain then receives these signals and interprets them as fear. The elicitation of emotions is thus viewed as a largely instinctive process, receiving little input from cognition. Evidence in favour of this theory includes an experiment by Strack et al.

(1988), where subjects viewed a cartoon while artificially inducing facial expressions. Subjects who held a pen between their teeth (resulting in a smile) judged the cartoon to be more amusing than those who held the pen between their lips (resulting in a frown).

The reverse position is adopted by the Lazarus theory (Lazarus, 1991), which argues that the cognitive appraisal of an event comes first, and then triggers the body's physiological response and subjective experience. Thus, emotional elicitation is viewed as a deliberative process driven by cognition. Animal behaviour studies have shown that there is a correlation between intelligence and emotionality (Ratner, 1989). Organisms that possess more sophisticated cognitive faculties exhibit richer and more varied emotional responses, and even the most intelligent animals do not express the full range of human emotions. For example, chimpanzees experience maternal attachments to their young, but show no observable signs of grief when their offspring die.

These approaches are not necessarily mutually exclusive, and some theories seek a compromise between the two positions. One early hybrid approach is the two-factor theory, which defines emotions as a combination of physiological arousal and cognitive interpretation. Bodily signals control the intensity of an emotion, while the different categories of emotional states (anger, happiness etc.) are a result of cognitive processes that take into account the context of the situation. This theory arose from an experiment by Schachter and Singer (1962) in which subjects were injected with adrenaline or a placebo and placed in a room with an experimenter who attempted to anger or amuse them. Those who were ignorant or misinformed of the drug's effects were more easily influenced by the experimenter's actions than the placebo group and those who had been informed. The results indicate that both arousal and environmental cues play a role in emotional elicitation.

Further evidence for a hybrid approach to emotional elicitation comes from studies on fear conditioning, the process by which organisms learn to fear certain stimuli. Experiments involving snake fear in primates have shown that the elicitation of fear has both learned and innate components (Ohman, 2003). Laboratory-bred infant rhesus monkeys initially exhibited no fear of snakes. However, when the infants observed video recordings of other monkeys reacting fearfully to snakes, they quickly acquired the same phobia. The same effect did not occur when the experiment was

repeated with other stimuli, such as flowers. This indicates that while learning is still required to associate snakes with the fear response, rhesus monkeys can more readily learn to fear snakes than flowers. Other studies have revealed that there is also an innate, evolved component to fear conditioning in humans.

2.2.1.3 Functions

The functional roles of emotions have been extensively investigated and debated. Authors such as Ekman (1992) argue that the primary function of emotion is to facilitate social interaction. Emotions are closely related to theory of mind, the ability to understand another's point of view, know their intentions and predict their actions. The expression of emotions can be considered a channel of communication that conveys social information in parallel with language-based exchanges. Emotional expression can evoke complementary emotions in others that reinforce social relationships. For example, expression of anger tends to evoke fear responses in others, while sadness elicits sympathy (Keltner and Haidt, 1999).

Social interaction is undoubtedly an important domain in emotion research, but emotions are also a source of adaptive behaviour at an individual level. A number of studies have investigated the effects of emotions on cognitive functions such as attention, memory and learning. It has been hypothesised that arousal directs attention towards emotion-eliciting stimuli, improving recollection of those objects, but impairing memory of peripheral details (Reisberg and Heuer, 2004). This theory is supported by studies of weapon focus, the tendency for eyewitnesses of violent crimes to focus their attention on the weapons held by criminals. Thus, witnesses often experience impaired recollection of other visual information in the scene, such as the perpetrator's face or clothing (Pickel et al., 2003).

Memory recall is modulated by a person's current mood (Parrott and Spackman, 2000). Moods are comparatively long-term affective states that are less object-specific and generally less intense than emotions. There are two mechanisms for this interaction. First is mood-dependent recall, the recall of memories that were stored at a time when a person's mood was similar to his or her current mood. Second is mood-congruent recall, the correlation between a person's current mood and memories that trigger similar emotions, regardless of the mood experienced at the time of storage. In an experiment by Snyder and White (1982), happy or sad moods were induced in

subjects, who were then asked to recall events from the previous week. Results indicated a correlation between the moods of the participants and the valence of the memories recalled.

Emotions can also bias reward processing in learning and decision-making (Baxter and Murray, 2002). This facilitates the rapid acquisition of behaviours necessary for survival in ancestral conditions. However it can sometimes have an adverse effect when emotions are applied to problems beyond those that they evolved to solve. For example, emotions can cause investors to acquire impulsive buying/selling behaviours that are detrimental to long-term success.

The affect-as-information model regards the conscious perception of emotions and moods as a source of information about non-conscious appraisal processes (Isbell, 2004). Positive emotions and moods indicate benign environmental conditions, increasing the brain's usage of heuristics such as stereotypes to guide information processing. Negative affective states indicate potential problems, reducing the usage of previously-acquired categorical information. Evidence supporting this theory includes an experiment by Isbell (2004), where subjects were given stereotype information about an individual (e.g. classifying her as an introvert or extrovert), accompanied by a story about the individual that partially conflicted with the stereotypes presented. Judgements of the individual by participants who reported a happy mood were influenced more by the stereotype information than those who reported an unhappy mood.

Overall, the vast amount of conflicting evidence shows that the problem of describing the interactions between emotions, cognition and behaviour in humans and animals is immensely complex. No single theory is likely to be adequate for all domains of emotion research. However, one approach that may prove useful in the computational domain is to model emotions as modulations of cognitive processes, as proposed by Dörner and Hille (1995).

2.2.2 Neurobiology Perspective

Much of the research on the neurobiological basis of emotions involves analysing the behavioural impairments of humans or animals with lesions (damaged tissue) in the brain area under scrutiny. Impairments to a specific emotional function may indicate

that the damaged structure normally controls that function. However, sometimes the structure merely acts as a connecting pathway between other brain regions that are more heavily involved in the process. Chemical or electrical stimulation of the brain and neuroimaging techniques such as positron emission tomography (PET) and functional magnetic resonance imaging (fMRI) scans can provide further evidence for or against a brain structure's functional role. Nevertheless, results in this field are often inconclusive, and many aspects of the brain's emotional circuits remain unclear.

Viewed from an evolutionary perspective, brain regions can be grouped into different components by the order in which they appeared as the brain evolved. The first component, the archipallium (primitive brain), contains structures that reside in the brain stem. Next is the paleopallium (intermediate brain), which contains the limbic system, a group of structures that historically has been regarded as the emotional centre of the brain. The most recent component, called the neopallium (rational brain), or neocortex, is utilised most extensively by higher mammals such as apes and humans, and comprises the brain's outer layers.

Historically, a clear distinction was drawn between the emotional brain, represented by the limbic system, and the cognitive brain, represented by the neocortex. The limbic system is a group of inner brain structures that includes the amygdala, hippocampus, cingulate gyrus, thalamus and hypothalamus. Emotions were thus regarded as evolutionary holdovers from our more primitive ancestors, epiphenomena of legacy systems that are no longer necessary in modern humans. However, the idea of the limbic system acting as the brain's 'emotion centre' has been criticised in recent years as being overly vague and simplistic (LeDoux, 1996). The limbic system's various components do not contribute equally to emotional functions, and certain structures outside the limbic system have been shown to possess key roles in emotional behaviour.

One brain structure that has received significant attention from researchers is the amygdala. There are actually two amygdalae – one in each brain hemisphere. In general, the left amygdala processes verbal stimuli, while the right amygdala processes visual stimuli (Buchanan & Adolphs, 2002). The amygdala contributes to the ability to make value judgements about other individuals, as evidenced by lesioned patients who suffer impairments to this function (Davidson, 2001). It also influences the formation of emotional memories (Buchanan & Adolphs, 2002).

Patients with lesions in this area experience more difficulty remembering events corresponding to a high emotional intensity than normal humans, while their memory of normal events remains unchanged. In contrast, damage to the hippocampus, an adjacent brain structure, results in uniform degradation of memory performance.

The amygdala plays a crucial role in fear conditioning, and damage to the amygdala can result in overly placid behaviour in the presence of danger. Lesions in this area have also been shown to impair the recognition of facial expressions of fear, but not other emotions (Davidson, 2001). Two distinct fear circuits are linked to the amygdala (LeDoux, 2000). One receives fast, reactive signals from the thalamus, allowing an organism to respond quickly to immediate threats. The other consists of slower, more refined signals received from the cortex, enabling the organism to act appropriately to facilitate long-term survival.

Another focal area for emotion research is the hypothalamus, a structure responsible for maintaining homeostasis, or keeping the body's physical parameters within acceptable margins. It is linked to many of the physiological responses associated with emotions, controlling blood pressure, heart rate, thermal regulation, sexual arousal, hunger and thirst. It also indirectly contributes to the formation of emotions by providing feedback from these physiological responses to other brain structures. In an early experiment, removal of the hypothalamus from cats resulted in a phenomenon called sham rage, where even in the absence of an external stimulus they exhibited the physiological symptoms of extreme anger (Bard, 1928). Later experiments demonstrated that electrical stimulation of certain regions of the hypothalamus could artificially induce anger and fear responses (Hess, 1957).

A brain structure that is believed to play a key role in the interaction between emotions and cognition is the prefrontal cortex. This structure performs the brain's executive function, prioritising between conflicting goals and modulating social behaviour. It plays a role in anticipatory emotions such as anxiety (Simpson et al., 2001). Lesions in the prefrontal cortex can impair the ability to anticipate future positive or negative consequences of decisions (Davidson, 2001). The prefrontal cortex is closely related to personality traits, and damage to this area can drastically alter one's personality. This effect was first observed in the famous case of Phineas Gage, a railroad construction foreman who in 1848 survived after a metal rod passed through the front of his head.

Other brain regions that are known to participate in emotional functions include the anterior cingulate cortex, cingulate gyrus and nucleus accumbens. The anterior cingulate cortex has been implicated in the conscious awareness of emotions. The cingulate gyrus contributes to the coordination of external stimuli with positive emotional memories, and to emotional responses to pain. The nucleus accumbens is linked to positive emotions associated with pleasure, reinforcement learning and addiction.

An ever-expanding list of brain regions linked to emotions reveals that just as there is no single brain structure controlling consciousness or memory, there is no single structure controlling emotions. Nor are structures within the brain neatly divided between those responsible for cognition and those responsible for emotions. Rather, the two processes are closely entwined and mutually dependent.

Fellous (1999) proposes an alternative biological mechanism for this relationship in the form of neuromodulators. Similar to neurotransmitters, neuromodulators are substances released by neurons that activate or suppress other neurons. However, while most neurotransmitters only pass information between two neurons, neuromodulators persist in the brain for significant periods of time and modulate neural activity over a relatively large area. Since emotions are also sometimes regarded as modulatory in nature and possess the properties of temporal persistence and diffuse action, Fellous suggests that they can be represented as dynamical patterns of neuromodulations. This theory provides a neurobiological foundation for the hypothesis that emotions are modulations of cognition and action.

2.3 Computational Models of Affect

Computational studies focus on developing models of affective processes, either to improve our understanding of biological emotions, or to enhance the performance of autonomous agents. These models can range from abstract, general-purpose architectures to highly-focussed task-specific implementations. There are significant limitations to each approach. Architecture-level models tend to be too broadly defined to be readily applied to specific tasks. Conversely, task-level models can be difficult to adapt to domains beyond those for which they were intended. Affective models outside of the robotics domain are often large-scale general-purpose AI frameworks

that emphasise cognitive roles of affect such as goal prioritisation and memory management (Aylett, 2006). These types of models typically favour cognitive appraisal theories of emotion such as those proposed by Ortony et al. (1988) and Lazarus (1984).

Sloman et al. (2005) regard many emotions as side-effects of the interactions of complex cognitive systems, analogous to the 'thrashing' that occurs during disk maintenance, rather than explicitly programmed discrete states. Sloman's CogAff schema is an attempt to develop a cognitive architecture that possesses the foundations necessary to exhibit affective behaviour implicitly. Three architectural layers are modelled by the CogAff schema: reactive mechanisms, deliberative reasoning and meta-management. A preliminary implementation of this model is being developed, called H-CogAff. Nevertheless, as with any highly abstract and incomplete description of mental processes, its implementation on a robotic platform would pose considerable challenges. Translating high level block diagrams and descriptive language into task-specific robotic algorithms is a non-trivial task.

Approaching the problem from a similar angle, Minsky's Emotion Machine model (Minsky, 2006) makes little distinction between emotional and cognitive processes, instead regarding individual emotions as different configurations of cognitive resources. To Minsky, words such as consciousness and emotion are merely 'suitcase-words' describing processes that are too complex to understand as a whole. Rather than seek simplified explanations for such phenomena, the model attempts to decompose them into highly specialised subsystems. Minsky's cognitive architecture contains six layers of increasing abstraction (Figure 2.2). Like most architecture-level models, the Emotion Machine model is a broad description of the mind, currently lacking the detail necessary for software implementation.

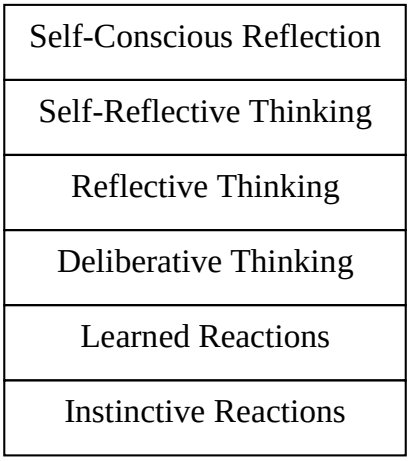

Figure 2.2: The Emotion Machine model's six processing levels.

A number of broad, but shallow computational models of emotion have been developed based on the ideas of Ortony et al. (1988), including the Oz Project's Em module (Bates, 1994) and the Affective Reasoner (Elliott, 1994). These models

attempt to improve the believability of autonomous agents by incorporating large numbers of simple emotions and behaviours. Emotions are divided into mutually exclusive categories depending on their eliciting stimuli (events, agents or objects). Appraisals are largely cognitive, but the influence of emotions on cognition is very limited. Instead, these models tend to focus on external displays of emotion and their effect on social interaction.

Grach and Marsella (2004) describe a generic model of emotionally influenced cognition, called EMA. The model is based on cognitive appraisal theories and utilises the Soar cognitive architecture. Emotions are generated from tables of appraisal dimensions (e.g. relevance, desirability, likelihood and causal attribution) for past and future events. The model regards coping as the inverse of appraisal – reducing the intensity of an emotion by attempting to restore the appraisal variables to an equilibrium state. This can involve changing the agent's actions or changing its beliefs about the situation. EMA is specified in fairly abstract terms, making task-specific adaptations difficult, and it is currently more suited to high-level tasks than real-time control of mobile robots.

One computational model that is closely aligned with our goal is Dörner's 'Psi' model (Dörner and Hille, 1995), which utilises emotions to modulate cognitive processes. According to Dörner (2003):

> "Affect and emotions naturally replace the knobs, dials, and switches on the 'control board' of an intelligent artefact."

Various global parameters are defined that represent broadly-encompassing aspects of information-processing systems, including:

- Arousal – Preparedness for action.
- Selection threshold – Aversion to choosing new goals.
- Resolution level – Precision of cognitive processes.

A set of basic needs are also defined:

- Existence preservation – Deals with obtaining nourishment and avoiding pain.
- Species preservation – Concerns matters of sexuality and reproduction.
- Affiliation – Represents the desire for social interaction.

- Certainty – A need for predictability of cause and effect.
- Competence – Success in problem solving and satisfying other needs.

Intentions are formed from these needs, and compete with each other for activation in a winner-takes-all manner. Global parameters are functions of the need values. For example, arousal is calculated from the sum of all needs. Emotions are implicitly represented as combinations of these parameters. Anger is represented as high arousal (‘tense’), high selection threshold (‘focussed’) and low resolution level (‘careless’). Anxiety arises from a combination of high arousal, low selection threshold (‘unfocussed’) and high resolution level (‘careful’). Joy emerges from medium arousal (‘moderately relaxed’), medium selection threshold and medium resolution level.

This model is being formalised in a software architecture called MicroPsi (Bach, 2003). MicroPsi is intended as a general-purpose architecture, and few task-specific implementation details or results have been presented.

Approaching the representation of emotions in a similar manner, but from a neurobiological perspective, Parussel (2006) implemented a model of artificial neuromodulations in a learning agent. Middle-layer neurons are assigned inhibitory or excitory receptors that modulate their sensitivity to inputs and/or probability of firing in response to modulators secreted by input neurons. Like Dörner, Parussel does not explicitly define any specific states or processes as emotions, but argues that they are inherently related to these modulations. This approach was tested in simple experimental gridworlds, where agents were evolved to maintain two resources (‘energy’ and ‘water’). The learning performance of modulating agents compared favourably to that of equivalent non-modulating agents.

Our model is partially inspired by the idea of representing motions as modulations of cognitive processes described in Dörner’s and Parussel’s models, but there are many points of difference. Unlike these models, in which emotions emerge implicitly from modulations, emotions in our model are explicitly defined and trigger parameter modulations. The application of our model to a wheeled mobile robot also differs from Dörner’s general-purpose cognitive model and Parussel’s relatively simple gridworld experiments. The differences are further elaborated in Section 3.2.

2.4 Robotic Affect

Mobile robots generally lack the high-level capabilities to benefit from the general-purpose cognitive models of affect often favoured outside of the robotics domain. In particular, most robots do not possess object recognition capabilities. Their sensors are often limited to distance measurements on a 2D plane. Furthermore, even if they possess sophisticated sensors such as onboard vision systems, the problem of real-world object recognition has not been solved in the general case.

Instead, robotic implementations tend to favour neurophysiological models, and typically employ emotions as internal 'sensors' that influence action selection (Aylett, 2006). These models are often inspired by somatic theories of emotion such as that presented by Damasio (1994). One of the main functions of this type of affect representation is to motivate a robot to respond quickly to certain events without waiting for its slower cognitive processes to ponder the situation. Thus, affect is regarded as a potential replacement for deliberative processing in robotic controllers. Interactions between affect and deliberative processing have received little attention in the robotics domain, because they are often viewed as competitors for the same role. Nevertheless, deliberative motion planning is an essential component of many hybrid mobile robot controllers, and performance would likely suffer if it were replaced with a reactive affect model.

Arkin (2005) defines two important roles for emotions in robots:

- *Social interaction* – Simulated emotions can enable robots to behave in a socially appropriate manner when interacting with humans.
- *Adaptive Behaviour* – Artificial emotions are a source of adaptive behaviour that can potentially improve a robot's general performance.

2.4.1 Social Interaction

The robotics community has focused largely on the social aspects of emotions. In this domain, a robot's emotions are generally clearly observable in the form of facial expressions, body language and/or tone of voice. While some researchers utilise biologically inspired models, many regard the socially relevant effects of emotions as more important than the underlying model. Focal areas include the appearance of

emotions, the subjective evaluations of humans who interact with emotional robots, and the influences of human emotions on robotic behaviour and learning.

One of the most widely-known examples of emotion-driven sociable robots is the ‘infant’ head Kismet developed by Breazeal (2003). Kismet’s control architecture is partially based on Valásquez’s Cathexis model (Valásquez, 1997). It incorporates affective drives as intensity values that are maintained close to a desired operational point, and within a bounded range of that point (called the homeostatic regime). A drive that exceeds an upper threshold becomes overwhelmed, necessitating a decrease, while a drive that falls below a lower threshold are underwhelmed, necessitating an increase. Kismet has three drives: social (a need for human-robot interaction), stimulation (a need to be stimulated by toys) and fatigue (a need for rest). Inspired by Damasio’s somatic marker hypothesis (Damasio, 1994), affective appraisals are performed that consider the drive intensities and perceived actions of humans or objects interacting with the robot in order to tag stimuli with several dimensions of affective information (valence, arousal and stance). The associated emotional responses (fear, anger, joy, sorrow, surprise, disgust, interest and calm) compete for activation in a winner-takes-all manner. In addition to influencing behaviour selection, these basic emotions are portrayed as variations in Kismet’s facial expression (Figure 2.3), gaze direction and tone of voice.

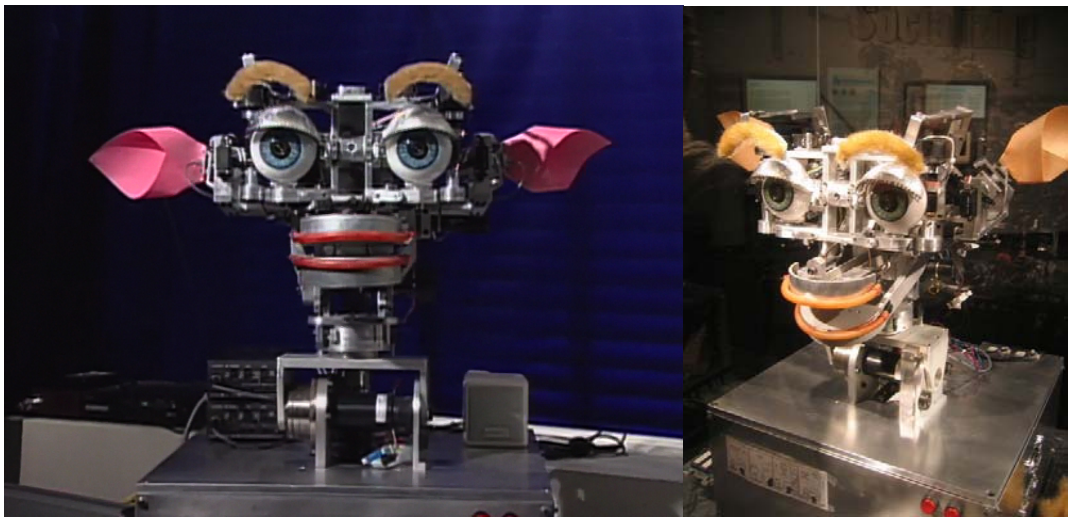

Figure 2.3: Kismet, the robotic infant.

The robotic head MEXI (Esau et al., 2007) employs a similar approach, combining visual and speech-based emotion recognition capabilities with an internal emotion

model that controls the robot's behaviours, facial expressions and speech utterances. This emotion model attempts to increase the value of its single positive emotion (happiness), while decreasing its negative emotions (anger, fear and sadness). It also attempts to maintain MEXI's drives at optimal (homeostatic) levels. A behaviour-based architecture is employed that incorporates both cooperative behaviour coordination and competitive winner-takes-all behaviour selection. Emotions and drives are utilised to control the gains assigned to different behaviours, affecting their relative contributions to motor control.

An emotion-capable social interaction architecture has been developed for the humanoid robot ISAC (Peters et al., 2001). The architecture includes a model of the person with whom the robot is interacting (the human agent), and an internal representation of itself (the self agent). Human emotions are represented as a single parameter, satisfaction, obtained from words with emotional content spoken by the person. Separate short-term and long-term memory structures are employed, potentially allowing the robot to model mood-congruent memory effects. The presented implementation is incomplete and no results are presented.

Arkin's TAME architecture (Arkin, 2005) models traits, attitudes, moods and emotions for behaviour-based robotic systems. Each category of affective state is represented by multiple dimensions of intensity values. Traits are based on the Five-Factor Model of Personality (McCrae and Costa, 1996) and consist of constant behavioural biases along the dimensions of openness, agreeableness, conscientiousness, extraversion and neuroticism. Attitudes are learned positive or negative biases associated with specific objects. Moods are long-term stimulus-independent states. Emotions are represented by six intensity values corresponding to Ekman's list of basic emotions (anger, fear, happiness, sadness, disgust and surprise) (Ortony and Turner, 1990). Traits and emotions have been partially implemented on a Sony AIBO robotic dog (Figure 2.11) for a range of tasks intended to entertain human observers (Moshkina and Arkin, 2005). In this implementation traits control global parameters, whereas emotions drive action selection. Both are hard coded and task dependant. Results consist of feedback from observers on their subjective evaluations of the robot.

Hollinger et al. (2006) utilise an emotion model based on the Mehrabian PAD temperament scale to generate robotic behaviours that appear emotional to human

observers. Emotions are represented as continuous regions on a three dimensional graph (the dimensions are pleasure, arousal and dominance), and control the selection of movement and dialogue responses. The focus of this research is on producing believable emotional responses rather than the detail of the model itself, and the results primarily consist of feedback from onlookers at the AAAI '05 Open Interaction Challenge.

Broakens' EARL framework (2007) utilises emotions detected in human observers as positive or negative rewards for reinforcement learning. This improves the learning performance of a simulated robot in a continuous gridworld environment. The framework also incorporates other social emotion-learning interactions, but few implementation details are provided: Detected human emotions can be utilised as additional state inputs, metalearning parameters (e.g. exploration/exploitation or learning rate) and social inputs to an internal emotion model. Artificial emotions can also provide rewards, state inputs and metalearning parameters. Finally, social feedback is facilitated by a real-world robotic head capable of portraying emotions as facial expressions.

2.4.2 Adaptive Behaviour

The problem of synthesising emotion-influenced adaptive behaviour in a non-social context poses considerable difficulties. The biological mechanisms underlying emotion are not well-understood or universally accepted, hindering attempts to create computational equivalents of these mechanisms. There is little consensus over what types of actions outside of the social domain should be regarded as emotional. It is also difficult to prove that any advantages provided by an emotion mechanism cannot be achieved by equivalent non-emotional systems.

While some authors have argued that artificial emotions can serve a useful role in robotics outside of the social domain, few successful implementations have been demonstrated. Some robots that utilise emotions as adaptive behaviours are nevertheless applied in a social context, interacting with humans or other robots. A majority of emotion-based robots employ very simple reactive control architectures, lacking the robustness of the hybrid reactive/deliberative architectures commonly employed in the general domain of mobile robot navigation. While the influence of

emotions on robot learning has been explored, few authors attempt to model the ability to learn to experience emotions appropriate to a given situation or environment. Quantitative results that demonstrate actual performance improvements due to affective processes are scarce, and those that exist tend to analyze the performance of an entire system; mechanisms or emotions are often not decoupled to analyse their individual contributions to performance.

Valásquez's Cathexis architecture (Valásquez, 1997), models Ekman's set of basic emotions (Ortony and Turner, 1990) as 'proto-specialist' agents (Minsky, 1986) executing in parallel. Secondary emotions are emergent, resulting from the blending of multiple basic emotions. Emotions are one of several inputs that control behaviour activation. Architectures based on Cathexis have been applied to a number of behaviour-based software agents and robots for a range of different applications.

Michaud et al. (2001) represent emotions as global background states of a hybrid deliberative-reactive architecture that change the robot's goals and modulate the parameters of its behaviours. Two dimensions are modelled: joy/sadness and anger/fear. Emotions are derived from changes in the energy levels of the robot's motives. These energy levels are an abstraction of the progress towards the motives' goals. The model has been utilised for a range of tasks, including human-machine interaction and autonomous recharging.

Murphy et al. (2002) utilise artificial emotions to facilitate adaptive behaviour and prevent deadlock situations in multi-robot tasks. Emotions are represented as discrete states that link progress towards goals to specific actions. The robots are heterogeneous, each possessing an independent set of task-specific emotions (e.g. happy, confident, concerned and frustrated) with different triggers and responses. For example, one robot triggers the happy state when it has no pending requests, and this state is tied to a "tell joke" response. Emotions are also linked to control parameters, most notably one controlling the robot's travelling speed.

Tingley and Browne (2006) employ emotions as abstractions consolidating multiple external inputs to directly control autonomous mobile robots in a multi-robot context. Five discrete emotions (curiosity, satisfaction, anger, motivation and tiredness) are modelled as hard-coded input-output mappings, each with different temporal and architectural characteristics.

Gadanho and Hallam (2001) employ a recurrent neural network model of emotions for behaviour coordination. This model makes a distinction between emotions and physiological responses, called feelings. Bodily sensations give rise to domain-specific feelings (e.g. hunger, pain and restlessness), from which a set of basic emotions (anger, fear, happiness and sadness) are derived. Emotions also influence feelings via an 'artificial hormone' feedback mechanism. The model provides reinforcement values to a learning algorithm that coordinates a set of hard-coded control behaviours. A subsequent version of the model (Gadanho, 2003) represents emotions implicitly in a goal system that interacts with a distinct cognitive system.

Neal and Timmis (2003) focus on a biologically-inspired model of a single emotion, timidity, in order to adapt a neural network-based reactive control system to different environments. The timidity mechanism is represented as an 'artificial endocrine system', which releases hormones dependent on obstacle proximity. Hormones directly influence the weights of the robot's neural network controller, increasing the robot's obstacle aversion in cluttered environments.

A similar artificial hormone mechanism is modelled by Avila-García and Cañamero (2005) to modulate perception and action selection of behaviour-based mobile robots. The control architecture contains a small number of simple behaviours (e.g. avoid, warm-up, feed and search) coordinated by a motivational state machine. Three motivations are defined for a robotic three-resource predator-prey scenario, each controlling a single physiological variable. These motivations are temperature (a desire to consume a static 'heat' resource), energy (a desire to consume a static 'food' resource) and integrity (a desire to avoid a dynamic 'predator', represented by another robot). Artificial hormones are represented as second-order modulations of the motivational system. Emotions are not explicitly modelled, but rather inferred through the robots' responses to the underlying hormone mechanism.

2.5 Adaptive Mobile Robot Control

In this thesis, artificial emotions are applied to a mobile robot controller in order to improve its adaptive capabilities. An adaptive controller can be defined as one whose parameters are automatically adjusted to suit the changing conditions encountered by the system being controlled. Thus, an argument could be made that most adaptive

controllers employ mechanisms that are on some level analogous to biological affect, even if they are not labelled as such. Nevertheless, it is worth mentioning some existing adaptive control approaches that are not directly inspired by emotions or related mechanisms.

A number of adaptive mobile robot controllers are based on potential field or motor schema approaches developed by authors such as Arkin (1989). Unlike earlier behaviour-based controllers such as Brooks' (1986) subsumption architecture, these controllers typically employ a cooperative behaviour coordination mechanism, where multiple behaviours simultaneously contribute to the robot's motor outputs. Behaviours are combined using a weighted vector summation, where the contributions of individual behaviours are controlled by their respective weights. Different control strategies arise from different combinations of continuous weight values. This approach is well-suited to adaptive weight selection via a secondary control layer.

Ram et al. (1992) employ simple case-based reasoning to adapting the control parameters of a schema-based reactive mobile robot controller to different environments. A set of discrete navigation strategies is defined, with each strategy corresponding to a specific gain configuration for the various behaviours. Some examples are 'clear-field' (apply high gain to goal-seeking behaviour and low gain to obstacle avoidance behaviour) and 'ballooning' (high obstacle avoidance gain). A given strategy is selected by matching it to the robot's detected environment. Performance of the adaptive controller compares favourably to a controller with static parameters. This approach is further developed by Santamaria and Ram (1997), employing TD(λ) reinforcement learning (Sutton and Barto, 1998) to map situations to parameter configurations.

Goel et al. (1997) describe an approach that automatically reconfigures the parameters of a schema-based controller to facilitate its escape from behavioural cycles caused by local minima. Rather than utilising random noise to escape from minima, this controller attempts to select appropriate modes of behaviour that are not susceptible to causing repetitive motion when subjected to the conditions detected. A model-based approach is employed to detect failures (repetitive cycles), perform causal analysis, and modify control parameters in response. Results from simulation experiments in randomly generated environments indicate that the controller can escape from the majority of local minima encountered.

Huq et al. (2008) present a fuzzy approach to modulating the behaviours of a schema-based reactive/deliberative controller. It addresses several problems that occur in other schema-based methods: becoming trapped in local minima, failure to travel between closely spaced obstacles, and oscillations in the presence of obstacles and narrow corridors. The controller incorporates both strategic schemas (global path planning using Voronoi diagrams) and reactive schemas (local obstacle avoidance and goal seeking). A fuzzy inference system generates weights controlling the contributions of individual schemas in a context-dependent manner. This method produces smoother motion and/or fewer collisions than an unmodulated controller (with all weights set to 1) in several real world experiments.

Beetz and Belker (2000) employ model- and test-based transformational learning to adapt the control parameters of a RHINO mobile robot navigation system. RHINO combines map-based path planning capabilities with the dynamic window obstacle avoidance approach (Thrun et al., 1997). Because it takes into consideration the robot's global environment and its kinematic and dynamic constraints, this is a more sophisticated and capable navigation system than purely reactive schema-based approaches. The adaptation approach defines a small set of diagnostic rules, or situations to which the controller should respond by adapting its parameters. These include close obstacles, narrow corridors, high target velocity and sonar crosstalk. Experimental results indicate that the learning approach improves performance over its original parameter configuration within a single real-world test environment. However, it is not tested in any other environments.

Each of these approaches differs from our own in its underlying controller and/or the functional roles of its adaptations. Unlike the majority of these methods, our controller is not based on weighted vector summations of schemas, but rather, weighted combinations of 360° vector fields and velocity windows, eliminating some of the problems caused by excessive data reduction (refer to Chapter 5 for details). The approach by Beetz and Belker (2000) is one exception, employing a dynamic window method similar to our velocity control layer. However, the functional roles of parameter adaptations in the two systems are dissimilar. Those of other approaches are also different from ours, to varying degrees. Most similar in motivation to ours is the model developed by Huq et al. (2008) – in particular facilitating a robot's escape

from local minima and enabling it to travel between close obstacles (e.g. narrow doorways).

2.6 Summary

The application of emotions and related affective states to robots is a multidisciplinary effort combining research from diverse fields such as psychology, neurobiology, artificial intelligence and robotics. A range of different perspectives on biological and computational emotions that have inspired this research were described. The current state of the art in robotic applications of emotions, affect and adaptive navigation were also discussed.

The next chapter begins the task of developing and implementing a model of robotic affect. Although it is inspired by various existing implementations, it differs from most of them in some key respects. Unlike the majority of research on robotic emotions, this thesis does not focus on social interaction. Instead, the question of whether emotions and affect can improve a mobile robot's adaptive performance in a non-social context is addressed. In particular, emotions are applied 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 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 this research they are utilised to augment the robot's deliberative capabilities by providing location-specific biases to path planning.

3 Overview of the Robotic Affect Model

To address the question of whether artificial emotions can improve a robot's performance beyond the social domain, we must first define these emotions in the context of our robotic system. This involves the development of a computational model inspired by theories of biological emotion. Ideally it will yield behaviours that appear non-deterministic[1], yet in the long-term are beneficial to the performance of a robot. Whether or not these behaviours are interpreted as emotions by human observers is of less importance, because the focus of this thesis is on mechanisms that provide practical utility rather than aesthetic appeal. Emotions in this framework are primarily abstractions that can aid in the design of intelligent systems. Hence, the model is intended to be useful from an engineering perspective, rather than merely scientifically interesting. Although inspired by biological theories, it is primarily a framework for modelling robotic emotions, not human emotions.

Human emotions do not exist in isolation. They interact with a wide range of cognitive processes, as well as other 'emotion-like' states such as feelings, drives, moods, attitudes and personality traits. Although these related concepts are sometimes described by the word 'emotion', the term 'affect' encompasses the model as a whole, and this framework includes stimuli, drives, moods and emotions as distinct, interacting components.

3.1 Robot Control Architecture

Affect models are inextricably tied to the underlying computational architectures that support them (Sloman et al., 2005). Our robot's implicit cognitive system is represented by the hybrid reactive/deliberative control architecture shown in Figure 3.1. Higher architectural levels have a supervisory role, providing loose goals that can be obeyed or overruled by lower level processes as the situation dictates. This allows the robot to benefit from the guidance of deliberative planning, while maintaining the real-time responsiveness of reactive control. Three control levels are implemented, each formulating the navigation problem in a different conceptual space. A

[1] A non-deterministic system is less likely to become trapped in repetitive loops. Emotions are arguably no less deterministic than other cognitive processes, but they do introduce additional complexity to the behaviour of an intelligent system, which is likely to make them appear non-deterministic.

deliberative layer is applied in position space, while two reactive layers operate in direction space and velocity space. These are similar to the deliberator, sequencer and controller layers in standard three-layer architectures (Gat, 1997), except the central layer (directional control) includes elements of both a sequencer and controller.

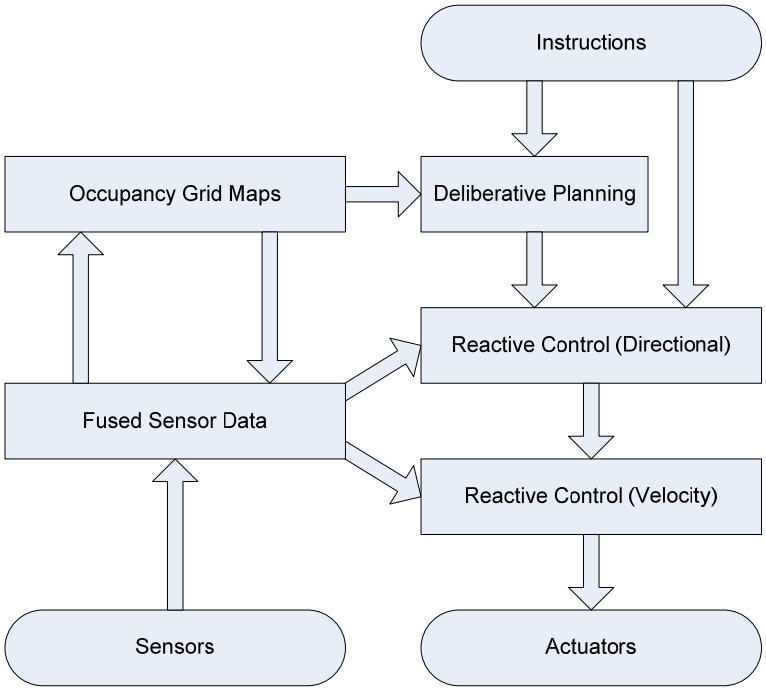

Figure 3.1: A representation of the robotic planning and control architecture.

This architecture is obviously much simpler than human and animal cognitive architectures. Computational models of cognition often include at least one reflective or meta-management layer (e.g. Sloman et al., 2005). Minsky (2006) hypothesises three reflective layers that represent different aspects of this process (reflective thinking, self-reflective thinking and self-conscious reflection). Our architecture does not include any explicit reflective layers, although affect does perform some performance analysis tasks normally associated with self-reflection.

The components of this architecture are also vastly simplified compared to their biological equivalents. The deliberative layer does not possess the ability to distinguish between objects. Instead, it conceptualises the environment in terms of discrete cells that are assigned attributes such as free/occupied and explored/unexplored. Thus, emotions can be associated with environmental locations, but not specific objects. The reactive layers have no predictive ability and limited

sense of location, instead relying on instantaneous sensor data and/or local representations to choose the robot's actions immediately prior to their execution. A biological 'reactive' layer would be closer to our deliberative layer in terms of capabilities.

The reactive and deliberative world representations are loosely analogous to working memory and long-term memory, respectively. Representation data are shared bidirectionally between the different layers. The deliberative layer utilises local sensor data to update its global map, but representation information can also flow down the hierarchy. The reactive layers can utilise global map data for obstacle avoidance in addition to local sensor data, allowing the robot to avoid certain objects that its sensors cannot easily detect.

Action instructions can only pass downwards through the hierarchy, and they are filtered through each control layer in turn. Instructions are not selected by a competitive winner-takes-all mechanism. Instead, each architectural layer attempts to satisfy the requirements of multiple objectives simultaneously by optimising an objective function. Individual objectives such as obstacle avoidance, goal-seeking and path-following are assigned weights that control their degree of influence over the robot's actions. A highly weighted objective exerts a greater influence over decision-making than an objective with a low weight, but neither is excluded from the decision-making process. Other parameters such as constraints, thresholds and search space granularities also affect the behaviour of these objectives, and can tilt the balance in favour of different types of behaviours.

No single set of parameter values can yield optimal behaviour in all conceivable environments and situations. For example, in open environments, large safety margins may be preferable, coming at minimal cost to performance and decreasing the likelihood of collisions by encouraging the robot to give obstacles a wide berth. If those same values are utilised in a confined office-block environment, they may prevent the robot from traversing narrow doorways. Performance can potentially be improved by modulating the robot's parameters to suit its current situation and environment. Regarding the previous example, safety margins can be maintained at high values until the robot encounters a narrow doorway, at which point they can be temporarily lowered to allow the robot to traverse it.

Ideally, adaptive parameter modulations provide a robot with the ‘mindset’ necessary to complete the task at hand. The difficulty with this approach is in determining exactly how and when each parameter should be modulated. This is the primary role of artificial affect in our architecture. It is a set of abstractions by which the problem can be made tractable.

3.2 The Model of Affect

Most existing models of affect and emotion are either high-level symbolic architectures inspired by cognitive appraisal theories, or neurophysiological models inspired by somatic theories (Aylett, 2006). Neurophysiological models are typically favoured by robotics implementations, but most examples are ill-suited to the navigation approach described. Unlike other ‘emotional robots’, ours does not employ a simple behaviour-based architecture. The architecture developed for this research includes multiple layers and employs a more continuous approach to decision-making than most behaviour-based systems. Thus, emotions cannot simply be employed to help choose the robot’s actions or behaviours from a discrete list.

Some authors argue that emotion is neither a purely cognitive system, nor an entirely distinct structure that competes with cognition for activation. An alternative option is offered in the form of modulatory representations of affect (Fellous, 2004; Dörner, 1995). Rather than existing as discrete states that simply trigger behaviours or actions, emotions are assigned a broad influence over the manner in which cognitive processes are conducted. Fellous (1999; 2004) proposes a biological mechanism – neuromodulation – that could account for these types of interactions.

A promising example of this type of model is Dörner’s ‘Psi’ framework (Dörner, 1995; Bach, 2003), which represents affect using global parameters that modulate information-processing systems. These modulations help allocate cognitive resources to suit a given situation and reduce the computational complexity of problem solving. Rather than explicitly defining discrete emotional states, emotions emerge implicitly from combinations of parameters.

This approach to affect representation is well-suited to the problem of mobile robot control. Most robotic navigation systems have internal parameters that must be tuned manually, or by automatic optimisation methods such as genetic algorithms. If the

parameters are tuned in a specific type of environment, they may result in unsatisfactory performance when the robot is tested in different environments without retuning. A modulatory model of affect can potentially improve the adaptive performance of a mobile robot by adjusting its parameters to suit the conditions it encounters.

Modulatory emotions need not be represented implicitly, as in Dörner's framework. Rather than emerging from the modulations of global parameters, a discrete set of emotion intensities are defined, from which various internal parameter values are calculated. Thus, changes in emotion intensities result in parameter modulations, instead of modulations representing the emotions themselves. This is more of a design decision than a philosophical position concerning biological emotions. It allows discrete emotions to be designed to fulfil specific purposes, rather than characterising certain nebulous internal responses as emotions following the design of a system. In our model, each emotion modulates a number of implementation-specific internal parameters to bias an intelligent system towards behaviours that in the long term are anticipated to improve aspects of its performance.

A multi-stage model of affect is employed, utilising the robot's perceptual and representation data to continuously update its parameter values in real-time. Many components of this framework are application-specific in that they are intended to modulate the parameters of a hybrid reactive/deliberative motion controller, rather than a general-purpose cognitive architecture. However, the basic structure of the model could be applied to different architectures or applications.

An overview of the model is shown in Figure 3.2. A set of emotional stimuli are derived from a range of sensor and representation data. The stimuli are combined in different configurations to form a set of basic emotions. Emotions are also represented in spatial maps, so the current emotions elicited by the robot result from a combination of the current stimuli and a persistent memory of previous emotions perceived in its current position. These maps are passed to the controller as location-specific biases to path-planning. They also contribute to the temporal persistence of emotions, a function proposed by authors such as Rolls (1999). The combined innate/learned emotions control a set of drives, each of which is responsible for modulating a subset of control parameters. A layer of second-order modulations is also modelled in the form of moods, which are global states derived from

combinations of emotions over large timescales. They modulate the ranges over which control parameters can be varied by drives.

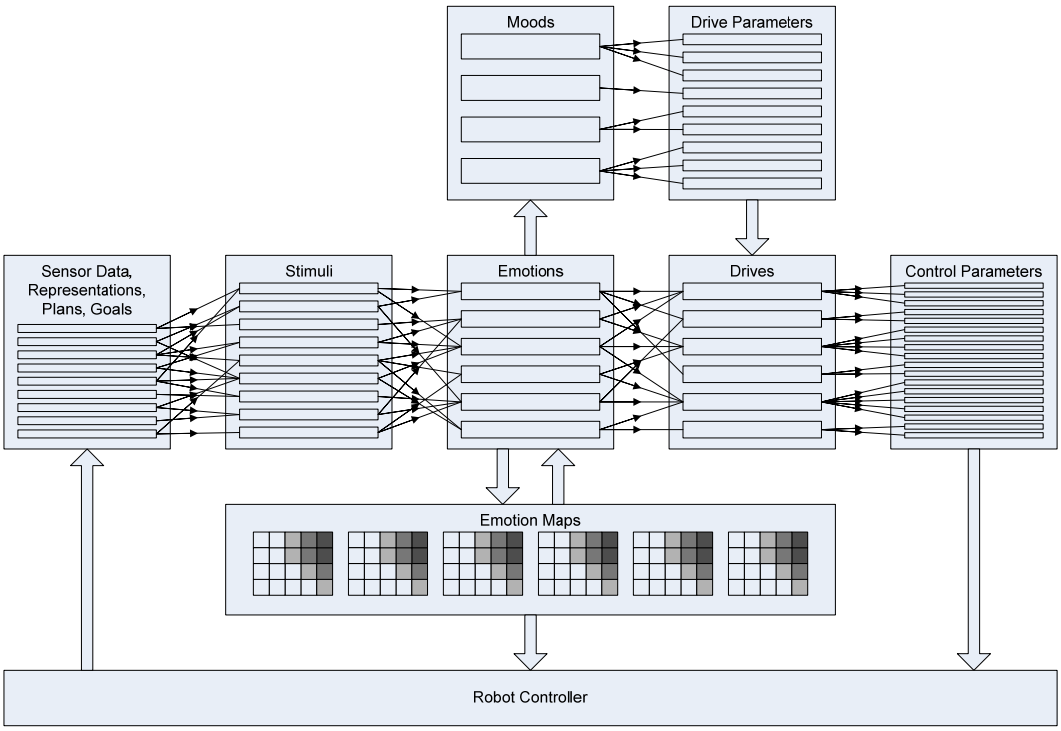

Figure 3.2: An overview of our model of affect.

3.3 Stimuli

Stimuli are internal or external events that trigger behavioural changes. In our framework, only stimuli that influence the robot's drives or emotions are included in this category. Although other stimuli are present, they are implicitly acted upon by the underlying cognitive system (represented by the navigation architecture in our implementation).

Somatic theories of affect (e.g. Damasio, 1999) regard physiological feelings such as pleasure, pain, skin temperature, blood-pressure, heart-rate, and muscle tension as important stimuli. The neurophysiological robotic emotion architecture EB (Gadanho, 2001) makes a distinction between feelings and the sensations (stimuli) that elicit them by giving feelings persistence in time via a hormonal feedback mechanism. Some biologically-inspired stimuli are included in our model (e.g. pain and fatigue). However, the majority of physiological feelings are not applicable to a mobile robot that lacks a biological organism's complex set of bodily senses.

Most affective stimuli in our model are simple, hard-coded cognitive interpretations of sensor and representation data, rather than bodily sensations obtained directly from sensors. Each stimulus is associated with a particular event or performance characteristic. Some stimuli are application-specific, and have no biological equivalents. The stimuli are:

- Danger – A collision is likely.
- Pain – A collision has occurred recently.
- Stuck – The robot's motion is repetitive, indicating an obstruction.
- Achievement – The robot is making satisfactory progress towards its goal.
- Density – The environment is densely occupied.
- Learning – World knowledge has increased recently.
- Mismatch – Internal maps and current sensor readings do not match.
- Error – Sensor data are inconsistent or anomalous, indicating a possible error.
- Cost – The best available path is unsatisfactory.

These stimuli are not modelled as binary states, but rather as continuous intensity values that grow and decay at different rates. The specific temporal characteristics of each stimulus are dictated by application-specific requirements. For example, danger grows and decays rapidly, allowing the robot to modulate its responses quickly to reduce the probability and potential consequences of a collision. Conversely, learning is slower to grow and decay, limiting the oscillatory motion that might arise from constant reprioritisation between exploration and goal-seeking.

3.4 Drives

Drives are often regarded as synonymous with motivations, needs or desires[2]. Drives are related to the concept of homeostasis, or maintaining bodily parameters within acceptable bounds. In the human brain this role is often attributed to the hypothalamus.

[2] Although these terms are sometimes used interchangeably, needs can be distinguished from desires by their importance for survival. Needs are survival-critical biological drives, such as hunger. Desires are less essential for survival, and thus are typically acted upon only when an organism's needs are satisfied.

TABLE 3.1: DRIVES

Drive	**Intensity**	**Responses**
Speed	Low	Move slowly, thereby reducing the rate of energy consumption, the probability of collisions, and the potential damage suffered during any collisions that do occur.
	High	Move quickly, improving the robot's ability to reach the goal position sooner.
Safety	Low	Reduce safety margins, allowing the robot to traverse narrow doorways or other openings.
	High	Maintain high safety margins, reducing the likelihood of collision.
Efficiency	Low	Choose a detailed computational style, potentially improving physical performance and/or collision avoidance.
	High	Choose a simple computational style, leaving more computational resources available for other tasks and/or conserving energy by reducing the CPU load.
Action	Low	Employ a hybrid reactive/deliberative approach to navigation.
	High	Disregard deliberative planning, relying on purely reactive control methods.
Introspection	Low	Rely on instantaneous sensor data for obstacle avoidance.
	High	Rely more heavily on internal representations than on instantaneous sensor data.
Exploration	Low	Assign low importance to the acquisition of new world knowledge.
	High	Favour knowledge acquisition over the exploitation of existing world knowledge.

Robots can possess both survival drives (analogous to needs) such as energy maintenance and 'informational' or strategic drives (analogous to desires) such as exploration (Dörner, 1995). A mobile robot navigation architecture can benefit from both types of drives, since it will typically consider both the robot's survival (e.g. by avoiding collisions) and the completion of its assigned task (e.g. point-to-point navigation or exploration). In the majority of models, drives elicit emotions, e.g. (Breazeal, 2003), or emotions are inferred from their interactions, e.g. (Dörner, 1995; Avila-García and Cañamero, 2005). Unlike these approaches, drives are represented

in the output stage of our framework, modulating cognitive parameters in response to explicitly-defined stimuli, emotions and moods.

Six basic drives are defined for this research. Each is responsible for modulating a distinct set of related cognitive parameters that influence the robot's decisions and actions towards or against a particular 'mode of behaviour'. The drives and their associated responses are given in Table 3.1. Although they are described in terms of low and high states, the drives are not represented as binary states. They are continuous intensities that control the positions of parameter values within a bounded range.

The first three drives (speed, safety and efficiency) can be categorised as survival drives, because they influence the robot's ability to avoid collisions and conserve energy. Conversely, the latter three (action, introspection and exploration) are strategic drives, because they influence general behavioural strategies and in most cases do not directly impact the robot's prospects for survival.

3.5 Emotions

Although the word 'emotion' has been used for a wide range of affective states and processes (Sloman et al., 2005), we include only a small number of core states in this category. According to Gunther (2004), emotions can be distinguished from other states and processes by several features:

- Intentionality – Emotions are directed at objects (either physical or conceptual), whereas feelings and moods are not directed at anything. For example, one is not merely angry, but angry at something. Our robot controller does not possess the ability to distinguish between physical objects in the world, so in our model emotions instead have 'intentionality' towards map locations.
- Cognitive preconditions – Emotions require certain cognitive beliefs about the world, whereas feelings occur independently of cognition. For example, fear requires the belief that a fear-inducing object is dangerous. In our architecture, these beliefs are represented implicitly within the stimulus functions.

- Charged valences – Emotions are intrinsically positive or negative, unlike other cognitive and motivational states. For example, joy and amusement are positive, while sadness and fear are negative. This property is represented by emotional influences on path planning in our system: emotions apply a positive or negative bias to the costs of map nodes in which they are elicited.

Our emotion model is inspired by theories of basic human emotions proposed by authors such as Ekman (1992) and Plutchik (2001), who argue that certain emotions are distinct and biologically innate, rather than cultural artefacts. However, not all emotional words necessarily represent distinct processes. For example, words such as 'terror', 'fear', 'dread', 'caution' and 'anxiety' may represent aspects of the same emotion. Thus, discrete categorisations typically define a small set of basic emotions, from which a range of secondary emotions are derived.

Different computational architectures support different classes of emotions (Sloman et al., 2004). Given the vast differences between human and robot architectures, there is little reason to strictly adhere to human-centric emotion categorisations unless human-machine interaction is the intended application. A number of robotic implementations favour human-centric categorisations such as Ekman's six basic emotions (e.g. Breazeal, 2003). However, when the goal is not simply to mimic human emotional responses, some basic emotions defined by psychological models are of limited practical value to the current generation of robots. For example, disgust is normally associated with taste, smell, cleanliness, aesthetic appeal and morals – concepts that are not relevant to the majority of robotic implementations. Conversely, some emotions that are not included in most human-centric basic emotion models may be of sufficient importance to represent as basic robotic emotions. Hence, the question should be, "Which emotions are potentially the most useful to our application?" rather than, "Which human emotions are considered basic?"

In our framework, emotions are an intermediary stage between stimuli and drives. Although some stimuli (e.g. fatigue) directly influence drives, most stimuli are translated into emotions first. The emotions and their associated stimuli and drives are shown in Table 3.2.

TABLE 3.2: EMOTIONS

Emotion	**Eliciting stimuli**	**Effects on drives**
Anger	High stuck, low achievement	Low speed, low safety, high efficiency
Fear	High danger, high pain	Low speed, high safety, low efficiency
Happiness	High achievement, low density	High efficiency, low exploration
Sadness	High pain, high error	Low efficiency, high exploration
Curiosity	High learning, low cost	High exploration
Surprise	High mismatch	High exploration, high action, high safety
Confusion	High error	Low action, high introspection

Anger arises when the robot perceives obstructions that may prevent it from achieving its goals. Typically, the main goal of our mobile robot is to arrive at a particular location, and the main impediments to achieving that goal are environmental configurations or dynamic obstacles that block its progress (e.g. narrow doorways, humans or other robots). Anger temporarily reduces the robot's safety margins in order to bypass an obstruction. Computational resources such as search resolutions and memory sizes are decreased, resulting in rougher, more 'careless' navigation. This is analogous to Dörner's 'resolution level' parameter (Dörner, 1995). Both of these modulations increase the likelihood of collisions, so the robot's speed is lowered to compensate.

Fear results from the prediction of potentially dangerous or painful events. In the context of mobile robot navigation, the main survival-threatening event is a collision. The most likely cause of a collision is a navigational 'mistake' resulting from sensor, actuator and/or software limitations. Thus, the greatest danger to the robot is itself. Our robot has no predators or prey, so there is no need for the computational equivalent of a fight-or-flight response. Instead, fear reduces the robot's speed, favours safety-enhancing behaviours, and devotes more computational resources to navigation. This reduces both the likelihood of collisions, and the potential damage resulting from any that do occur.

Happiness arises when the robot is successfully achieving its goal, and when the environment is perceived to be a low navigational challenge. Together with sadness, it

is assigned the role of modulating the overall levels of computational efficiency to match the difficulty of the task at hand. Trivial tasks such as the traversal of obstacle-free environments can be accomplished without compromising efficiency. Difficult tasks generally require the robot to make greater tradeoffs. Happiness also decreases the drive to explore the environment, since exploration is less advantageous if the robot already possesses sufficient world knowledge to successfully complete the task at hand.

Sadness is caused by potential loss represented by the pain and error stimuli. Overall, it results in increased computational effort, and it causes the robot to favour exploration of the environment over exploitation of existing world knowledge. This enables the robot to find alternative paths that may be superior to the one that resulted in sadness.

Curiosity not regarded as a basic emotion in most human-centric models, but in our model it is assigned the role of controlling the bias between exploration and exploitation. It is elicited when the rate of knowledge acquisition is high, and dampened by low quality planned paths. A high exploration drive can have an adverse effect on performance, particularly if the environment is already known, so this emotion allows the robot to favour exploration only when it is likely to yield additional world knowledge.

Surprise represents a mismatch between the robot's predictions and its perceived world. It can arise if the robot encounters an unexplored environment, or if it suffers a localisation failure. If the robot is in a known environment, hybrid reactive/deliberative navigation generally results in superior paths than purely reactive control. However, deliberative path planning requires accurate knowledge of the environment. Hence, when surprise is elicited, the robot increasingly favours reactive approaches to navigation over the hybrid reactive/deliberative method it normally employs.

Confusion is also excluded from the majority of basic emotion models. It occurs in the presence of internal inconsistencies caused by sensor limitations or errors. This represents a low level of confidence in the robot's perceptions, so abstract internal representations are favoured over sensor data. Confusion discourages the robot from

planning actions that previously resulted in collisions, even if its sensors currently detect no danger.

Location-specific intensity values near the robot grow or decay in response to each emotion's associated stimuli. The current intensity value of each emotion is a function of both its currently perceived stimuli and these location-specific intensities. The resulting emotion maps also yield positive or negative biases to the cost values of individual nodes during path planning. Happiness, curiosity and surprise[3] have positive valence, so they reduce node cost values, increasing the probability that the robot will plan a path through those nodes. Conversely, anger, fear, sadness and confusion have negative valence, so they increase costs, reducing the 'attractiveness' of nodes in which they are elicited. In this sense, the emotions are analogous to learning rewards and punishments applied to environmental locations.

3.6 Moods

Moods can be differentiated from other affective states and processes by their temporal characteristics – they persist for longer than stimuli, drives and emotions, but they are not as persistent as personality traits. They also have lower activation levels and intensities than emotions, and they are generic rather than object-specific (Moshkina and Arkin, 2005).

Few robotic implementations explicitly include moods as distinct affective states. One exception is the TAME (Traits, Attitudes, Moods and Emotions) framework (Moshkina and Arkin, 2005). Two mood categories are modelled in TAME: positive and negative affect (which are largely independent of each other).

Similarly, positive and negative moods are included in our model, as shown in Table 3.3. Moods increase when their associated emotions are highly activated for extended periods of time, and decrease when their emotions are inactive. They modulate the maximum or minimum values of survival drives (safety, speed and efficiency), affecting the level of influence emotions have over the robot's behaviour. Strategic drives (exploration, action and introspection) are not constrained by these moods. Nor do their primary eliciting emotions (surprise, confusion and curiosity) contribute to

[3] The valence of human surprise is not universally positive, but rather dependent on its triggering event (Kohler et al., 2003). However in our model it is assigned positive valence for practical reasons: to encourage the robot to explore map areas that do not match its sensor data.

the modulation of moods. The reason for this is related to implementation details that are described in Chapter 8.

If positive mood has a high intensity and negative mood has a low intensity, the robot is more likely to favour fast, safe and computationally efficient parameter values, while emphasising reactive rather than deliberative processes, and giving low importance to exploration. Conversely, if positive mood is low and negative mood is high, the robot increasingly favours slower motion, lower safety margins, and computationally-intensive processing styles. It also favours deliberative path planning over reactive control, and it is more likely to regard exploration as highly important. A high level of both positive and negative mood leads to 'bipolar' behaviour, where the robot oscillates between highly divergent parameter values depending on its momentary affective states. Low positive and negative moods lead to emotionally-muted behaviour, with emotions having lower overall levels of influence over the robot's decisions and actions.

TABLE 3.3: MOODS

Mood	**Activating emotions**	**Drive limits modulated**
Positive	Happiness	Speed maxima, safety maxima, efficiency maxima
Negative	Anger, fear, sadness	Speed minima, safety minima, efficiency minima

3.7 Summary

A computational model of affect has been developed that incorporates stimuli, drives, emotions and moods as distinct, but highly interconnected states and processes. The purpose of the model is to improve the general adaptive performance of an individual mobile robot, rather than to facilitate social tasks such as human-machine interaction. Unlike many robotic emotion models, this framework operates in conjunction with an independent cognitive system, represented by a hybrid reactive/deliberative mobile robot navigation architecture. Emotions and other affective states and processes modulate the decisions and actions of the cognitive architecture, but they are not the primary motivators of behaviours.

This chapter gave a high-level description of the robotic affect model. Before any further implementation details or results are presented, the mobile robot and experimental setup will be described (Chapter 4), and the robot's underlying planning and control architecture will be covered in more detail (Chapters 5-7).

4 Mobile Robot Hardware and Software

The planning and control architecture developed in this thesis is sufficiently generic that it can be (and has been) applied to a range of different wheeled mobile robots with few modifications (Roehr, 2008; Lee-Johnson et al., 2007). Nevertheless, it is designed with a specific robot in mind. Shown in Figure 4.1, the target platform is MARVIN (Mobile Autonomous Robotic Vehicle for Indoor Navigation), a custom-built mobile robot intended for security and public relations applications (Carnegie et al., 2004).

MARVIN's main task is to autonomously patrol indoor office block environments, recording individuals with an onboard camera and requesting identification. If an individual is authorised to be on the premises, the robot can guide him or her to a designated location. Conversely, if MARVIN determines that an individual is unauthorised, it can trigger an alarm. Ideally, the robot will also monitor its own power consumption, autonomously docking with a recharging station when it detects that its batteries are low. The focus of this thesis is not on these high-level tasks, but our navigation and control architecture provides some of the underlying capabilities required to perform them autonomously.

Figure 4.1: MARVIN.

The architecture is implemented on a simulated version of MARVIN. The robot's physical hardware is briefly discussed insofar as it influences the parameters and constraints of the simulated robot. In addition to the robot controller itself, software developed for this research includes a graphical user interface, robot simulator and a procedural environment generator.

4.1 Hardware

MARVIN's hardware can be divided into two main components: a base containing hardware required for navigation and high-level computation, and an upper body whose primary functions are to improve the robot's aesthetic appeal and facilitate human-machine interaction. Since this thesis focuses on the application of affect to mobile robot control rather than human-machine interaction, the upper body is of less relevance. Nevertheless, the torso and head unit does influence some design decisions, so it is briefly discussed for completeness.

4.1.1 Base Unit

A small form-factor PC (an Athlon XP 2000+ with 512 MB RAM) housed within the base unit is currently the robot's primary controller. The PC interfaces to the robot's hardware components via a National Instruments PCI-6229M data acquisition card (Cawley and Carnegie, 2006) and standard USB ports. As part of an ongoing hardware upgrade, graduate student Johnny McClymont is replacing this PC and DAQ card with a more powerful notebook PC and a custom microcontroller interface. Power is provided by two 12 V flooded lead-acid batteries in series. An ACE-828C DC ATX power supply converts the resulting 24 V input into signals appropriate for the PC and its peripherals (Lee-Johnson and Carnegie, 2003).

MARVIN possesses a two-wheeled differential drive system supported by casters at the front and rear. The wheels are driven independently by two 24 V DC permanent magnet brush motors, which are controlled by a pair of RHINO DS72K H-bridge motor drivers from Dynamic Controls Ltd. The motor drivers interface to the PC via a custom-built microcontroller board (Cawley and Carnegie, 2006).

Motor step functions obtained under load (i.e. driving the robot rather than freewheeling), with the motor drivers operating at 100% duty cycle, are shown in Figures 4.2 and 4.3. When its motors are driven at full duty cycle, the robot can move forward at approximately 1.6 m/s (although for safety reasons this speed tends to be impractical indoors). It takes around 2 seconds to accelerate and decelerate between a stationary state and these maximum speeds. The robot's acceleration/deceleration (shown in Figure 4.3 filtered by a moving average with a window size of 10 samples)

is not uniform throughout the duration of each transition, but its magnitude typically varies between 0.4 m/s^2 and 1.6 m/s^2.

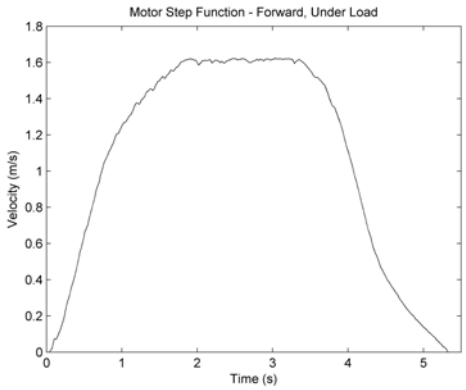

Figure 4.2: Velocity step function.

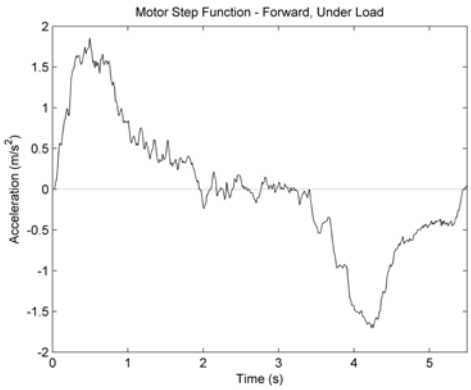

Figure 4.3: Acceleration step function.

Two HEDS-5700 optical encoders, each attached to a different motor, provide odometry information that can be utilised for velocity measurements and localisation. Like all odometers, these sensors result in cumulative localisation errors from sources such as conversion errors, update rate limitations, wheel slippage and non-constant wheel diameters, so they are only useful as a primary means of localisation over short distances. Long-term localisation requires that they be utilised in combination with exteroceptive sensors.

MARVIN's primary means of exteroception are Sharp GP2Y3A003K0F and GP2Y0A02YK infrared distance-measuring sensors. The GP2Y3A003K0F sensor comprises five separate pairs of infrared emitters and detectors, resulting in an overall coverage angle of 25°. It has a detection range of 40-300 cm. The GP2Y0A02YK is a single-beam sensor with a detection range of 20-150 cm. Voltage-distance relationships for each sensor type taken from their respective datasheets are given in Figures 4.4 and 4.5. Due to the nonlinear relationships shown in these graphs, changes in distance correspond to smaller changes in voltage as the range increases. Larger range measurements are therefore more susceptible to measurement errors and electrical interference, resulting in increased sensor noise. Any object closer than a sensor's minimum detection range gives an erroneous measurement, appearing to be further from the sensor than its actual distance. It is therefore necessary to position each sensor such that objects are unlikely to enter its dead zone.

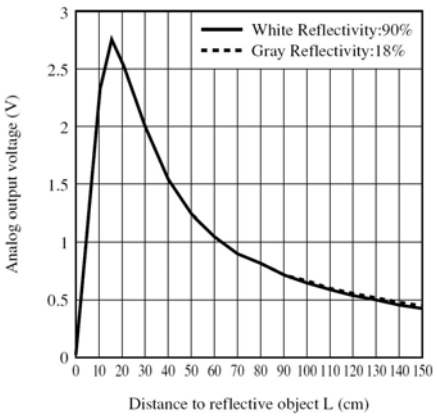

Figure 4.4: GP2Y0A02YK voltage-distance relationship.

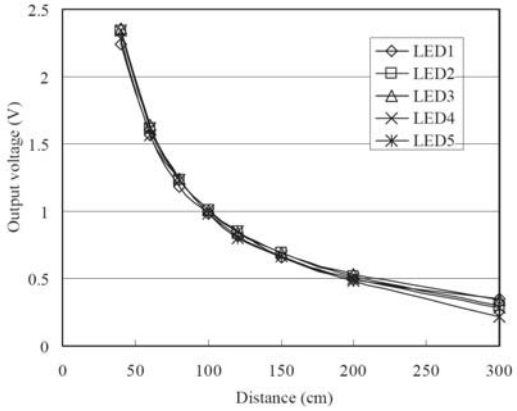

Figure 4.5: GP2Y3A003K0F voltage-distance relationship (dead zone not shown).

Ten GP2Y3A003K0F sensors are arranged in a ring around the top of the base unit, pointing outward from the robot's centre axis. The 40 cm dead zone is partially negated by attaching the sensors to the narrowest end of the base unit. Nevertheless, the remaining ~25 cm of dead zone is unsatisfactory. This problem is alleviated by attaching a ring of GP2Y0A02YK sensors directly beneath the GP2Y3A003K0F sensor ring (at an approximate height of 75 cm), pointing at a 45° angle downwards. The GP2Y0A02YK sensor's effective horizontal distance detection range becomes 14-75 cm, ensuring that most objects cannot enter its dead zone without having already collided with the outer edge of the robot's chassis. Any measurement over 75 cm indicates a floor surface lower than the surface upon which MARVIN currently resides. This allows the robot to detect uneven floor surfaces and dangerous drop-offs such as staircases.

4.1.2 Upper Body

MARVIN's upper body consists of a humanoid torso and head unit that can change its posture to portray a limited form of body language. Torso movements and extension of the head are driven by four Linak LA12 linear actuators controlled by custom-built H-bridge motor drivers (Carnegie et al., 2004). Other head motions are driven by three Hitec HS-815 servo motors. All actuator inputs are connected to a Mini SSC II controller, which interfaces to the PC's serial port. Attached to the head unit are a spotlight, speaker, microphone and webcam. The spotlight is controlled by the Mini SSC II, while the other peripherals connect directly to the PC.

The absence of an expressive face is a significant impediment to conveying sophisticated human emotions. Nevertheless, variations in spotlight intensity and voice parameters in combination with different head and body postures can portray certain affective states that are useful to an autonomous security guard. As shown in Figure 4.6, the robot can become physically larger and appear more intimidating to potential intruders by raising its head and tilting its shoulders forward. Conversely, by withdrawing its shoulders and head, MARVIN can adopt a more submissive posture in the presence of authorised individuals.

Figure 4.6: Example body postures (Carnegie et al., 2004) – (a) intimidating and (b) submissive.

4.1.3 Navigational Challenges

MARVIN's nominal width of approximately 70 cm can impede navigation in a confined office block or laboratory environment, where some doorways may be barely wider than the robot. The torso unit may exacerbate the problem by protruding beyond the width of the base unit in certain configurations. Its large height (up to around 1.8 m, depending on its head extension) can also cause problems, as the infrared sensors do not cover the length of the torso and head unit. Hence, the robot is vulnerable to collisions with obstacles that reside outside of the sensors' planes of detection (e.g. objects mounted high on walls). The infrared sensors can fail to correctly measure the distances of objects constructed from transparent materials (e.g. windows and glass cabinets). Additionally, the large bulk and weight of the robot may result in considerable damage (both to the robot and to its surroundings) in the event of a high-speed collision. MARVIN's controller must address some of these problems to ensure that it can operate safely in complex indoor environments.

4.2 Simulated Robot

For the purposes of robotics research, simulated robots have a number of advantages over their real-world counterparts:

- Downtime – Simulations can be executed any time, whereas real-world robots generally suffer long periods of downtime for recharging, maintenance or development. Robots are often shared between multiple projects, further limiting their availability. This is a significant problem for MARVIN, which is an active research platform undergoing continual development. As a result, MARVIN is frequently in a non-working state.
- Focus – In simulation, some robot capabilities can be omitted (e.g. localisation) in order to focus on the task under investigation (e.g. reactive control). In the real world, the robot will not function correctly unless all necessary capabilities are fully implemented.
- Supervision – Simulation-based experiments can be unsupervised for extended periods of time, whereas experiments on real-world robots generally require constant supervision and human input.
- Controllability – Simulated experimental conditions can be precisely controlled. Conditions are repeatable to an arbitrary level of precision, and they can be iteratively modified. In the real world, experimental conditions cannot be controlled or replicated as precisely.
- Speed – Simulation experiments can be completed faster than real-time. Real-world completion speed is limited by physical constraints.
- Safety – Many experiments that can be conducted in simulation are impractical in the real world for safety reasons. In particular, any experiments where the robot is likely to collide with people or objects are only feasible in simulation.

However, there are disadvantages to simulation-based research:

- Simulation accuracy – Simulations are only approximations of real-world systems, and are no substitute for real world experimentation when development surpasses the prototype stage. No simulation can model the full

complexity of dynamic real-world environments, especially those that contain unpredictable elements such as humans.

- Complexity – Given that simulations are generally less complex than real-world environments, the usefulness of some complex behaviours may be limited.
- Sensors – Some sensors are difficult to model in simulation (e.g. image-based sensors). However, MARVIN's primary navigation sensors are infrared rangefinders, which are relatively simple to simulate.
- Computational overhead – Simulations require additional overhead that could otherwise be spent on improving a robot's performance. However, this is not a drawback if the robot's onboard processor is sufficiently slower than the computer(s) running the simulation, as is the case for MARVIN.

The advantages outweigh the disadvantages in the majority of the research planned for this project. Thus, the decision was made to develop a simulated version of MARVIN and its environment that would serve as the primary research platform.

A robot's interactions with the world are summarised in Figure 4.7. Changing from a real-world implementation to a simulation requires the removal of sensor and actuator hardware interfaces, and the addition of software versions of the sensors and actuators. These interact with a simulated representation of the robot's environment.

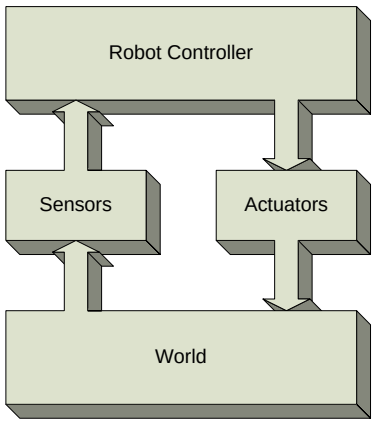

Figure 4.7: Robot-world interaction.

Robot pose and velocity variables are stored independently for both the controller and the simulator. When odometer errors are incorporated into the simulation, the robot's internal representations of these variables diverge from their actual values, as they do in a real-world robot. Similarly, the controller's internal representation of its environment is separate from the simulated representation. However, in the majority of experiments conducted for this research the problem of localisation is disregarded

so that we can focus on other issues[1]. Hence, cumulative odometry errors are typically set to zero, and only exteroceptive sensor errors are enabled.

4.3 Programming Environment

The robot control software and simulator are developed in MATLAB, a technical computing environment for numerical computation, visualisation and high-level programming. MATLAB programs can be developed rapidly compared to lower-level languages such as C++. MATLAB includes a built-in interpreter, so code does not need to be compiled before execution. Very little additional code is required to present data graphically onscreen. Its default data element is a dynamic matrix of 64-bit floating point numbers, and most functions are designed and optimised for matrix computations. An extensive library of specialised toolboxes is available for domains useful to robotics, such as data acquisition, control systems, optimisation and signal processing.

This high development efficiency comes at a cost to execution speed. MATLAB programs are typically much slower than C/C++ executables, particularly if they contain many loop structures. To compensate for this, functions written in C or C++, called MEX-files, can be executed from within MATLAB. MATLAB's built-in profiler is utilised to measure the relative execution speeds of each function within a program, revealing any performance bottlenecks. Bottlenecking functions are then rewritten in C as MEX-files, which generally yield orders-of-magnitude performance gains. Generally, there are sufficient computational resources at the robot's disposal that we can focus on maximising its physical performance rather than its computational efficiency.

Another disadvantage of MATLAB is its limited multithreading capabilities. While recent versions provide support for multi-core CPUs, distributing matrix computations across multiple processors, MATLAB programs cannot readily execute multiple loops concurrently. This limits its performance in some robotics applications, where different sensors, actuators, communications functions and architectural levels ideally should be updated at different frequencies. However, this is not a significant

[1] This is expected to have a non-trivial influence over the behaviour of the system. For example, because the robot's position is assumed to be accurate, internal maps and deliberative control techniques become more useful, and thus the robot can be less dependent on reactive control approaches.

drawback for MARVIN, whose sensor and actuator updates can be performed independently by the DAQ card and/or microcontrollers.

4.4 Graphical User Interface

MARVIN's pose, sensor readings, internal data and environment are displayed in a MATLAB figure window (Figure 4.8). Graphical objects are created using MATLAB's *patch* and *line* functions, and their properties are modified using *set*. This is more computationally-efficient than deleting objects and recreating them each time their vertices change.

Pose (position and orientation) is represented by a circle of the robot's radius and a direction arrow. Three pose objects can be displayed: estimated pose, actual pose, and target pose. These objects are mouse-movable. Position or direction can be adjusted by clicking and dragging the circle or direction arrow, respectively.

Infrared sensor beams are displayed as lines extending from the sensor locations relative to the robot's actual pose. Estimated obstacle positions are represented as points relative to the robot's estimated pose. Planned paths are displayed as dashed lines connecting map nodes.

The actual environment is represented by polygons and lines, whereas the robot's internal representation of the environment is an occupancy grid map drawn as square nodes of varying intensities. Only nodes within distance d of the robot's estimated pose are drawn on-screen because it is very slow to redraw the entire map each screen update.

Clicking the **Manual/Auto/Batch** button toggles between three main modes of operation (when it is toggled the button's text label changes to represent its new state). Manual mode allows the user to move the robot's estimated, actual and target pose objects, and the robot will carry out this single instruction. Auto mode executes a pre-programmed sequence of instructions. Batch mode can execute multiple instruction sequences, iteratively adjusting experimental environments or robot control parameters for tuning or testing purposes.

The main program begins executing when the user clicks the **Start** button. A timer callback is triggered that periodically executes a new control cycle until the

instruction is completed or the user clicks the **Pause** or **Stop** buttons. The robot's default control period is 100 ms, but the program can execute faster than this real-time value. During most simulation experiments the timer is configured so that it executes the next control cycle as soon as the previous one is completed.

Figure 4.8: MARVIN's graphical user interface.

The screen update rate can be adjusted by controlling the interval between *drawnow* function calls. More frequent updates result in smoother onscreen motion, but lower overall execution speed. Forced screen updates (*drawnow*) are only necessary when the program is running at faster than real-time, because the screen will automatically update when MATLAB's event queue is empty.

4.5 Program Structure

Figure 4.9 shows a block diagram of the robot software. When operating in manual mode, the program cycles through a single control loop until the robot converges on its goal point. In auto mode, this loop is expanded to incorporate additional steps, where upon completion of one navigation instruction, another is automatically

triggered. Batch mode further expands it to load new environments and/or update parameters when an experimental iteration has completed. This is analogous to a set of nested *while* loops, except that each new control cycle is triggered by a timer.

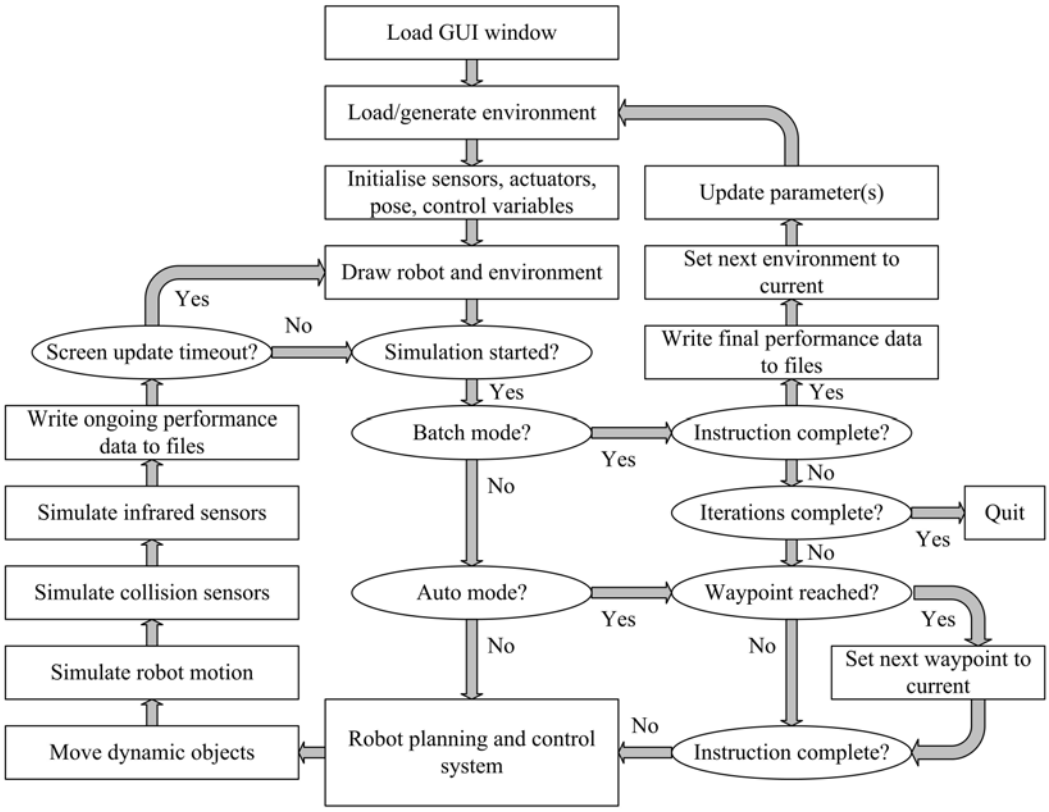

Figure 4.9: Block diagram of program structure.

4.6 Simulated Environments

Simulated environments are represented by 2D polygonal maps containing objects stored as sets of vertices (linked corner coordinates). Simulated objects can have a number of different properties that affect how they behave, and how easily they are detected by the robot's sensors:

- Static – Objects with this property do not move, and they are not affected by the robot if it collides with them. Walls, doors and most furniture have this property.
- Moveable – These objects do not move unless the robot makes physical contact with them, at which time they are pushed away from the robot. A moveable object can only be pushed if the direction of movement is not

obstructed by another object. Light objects that are not attached to a static surface (e.g. boxes and rubbish bins) possess this property.

- Dynamic – Objects that move of their own volition such as humans or other robots have this property. Their movements are also limited by collision detection with other objects.
- Limited visibility – These objects are undetectable, or only partially-detectable by the robot's sensors. They may be transparent or opaque to infrared light (e.g. glass cabinets) or sparsely-occupied (e.g. fences), or they might primarily occupy an area outside of the robot's plane of detection (e.g. narrow-legged tables).
- Floor – These represent floor drop-offs such as staircases. They are not detected by the robot's primary set of infrared sensors, but the downward-directed sensors can detect them.

4.6.1 Procedural Environment Generation

Simulated environments can be created manually in order to measure or train the robot's behaviour in highly controlled conditions. However, if a controller is designed or trained for optimal performance in a single environment, it is likely to perform poorly in other environments. The robot's general performance can be more effectively gauged and improved by utilising a range of different environments. Rather than construct a large number of environments manually, which is very time-consuming, random indoor environments are generated procedurally.

The procedural environment generation algorithm is shown in Figure 4.10. A tree structure is constructed that contains rooms of random dimensions connected by randomly-placed doors. Rooms are recursively added to the structure until there are no unconnected doors remaining.

The environment is segmented into blocks representing the smallest room size. The default block size is 2×2 m. A grid map containing the room index associated with each block is constructed in parallel with the tree structure. Rooms are categorised as either corridors, which are one block wide, or offices/laboratories, which are at least two blocks wide. A room's category can affect its object placement rules.

If a room of the selected dimensions cannot fit in the available space, rooms of decreasing size are compared with the available grid map spaces until one is found that fits.

```
Place room:
    Randomly select room dimensions.
    If room does not fit,
        Try successively smaller rooms until one fits.
    Determine maximum number of doors to attach to room.
    Repeat until all doors are placed or rejected,
        Randomly select door position.
        If block adjacent to door position is not already occupied by a room,
            Place door.
            Place room (recursion).
        Else,
            Reject door position.
Place extra doors:
    Repeat n times,
        Randomly select door location from available list.
        Update adjacent rooms.
Generate wall vertices:
    For each room,
        For each wall,
            Get wall and door endpoints.
            Generate vertices for connecting rectangles.
Place obstacles:
    Repeat until list of available blocks is empty,
        Randomly select block from list.
        Randomly select object type.
        Randomly select object dimensions.
        Generate object vertices.
        Remove adjacent blocks from list.
```

Figure 4.10: Pseudo-code representation of the procedural environment generation algorithm.

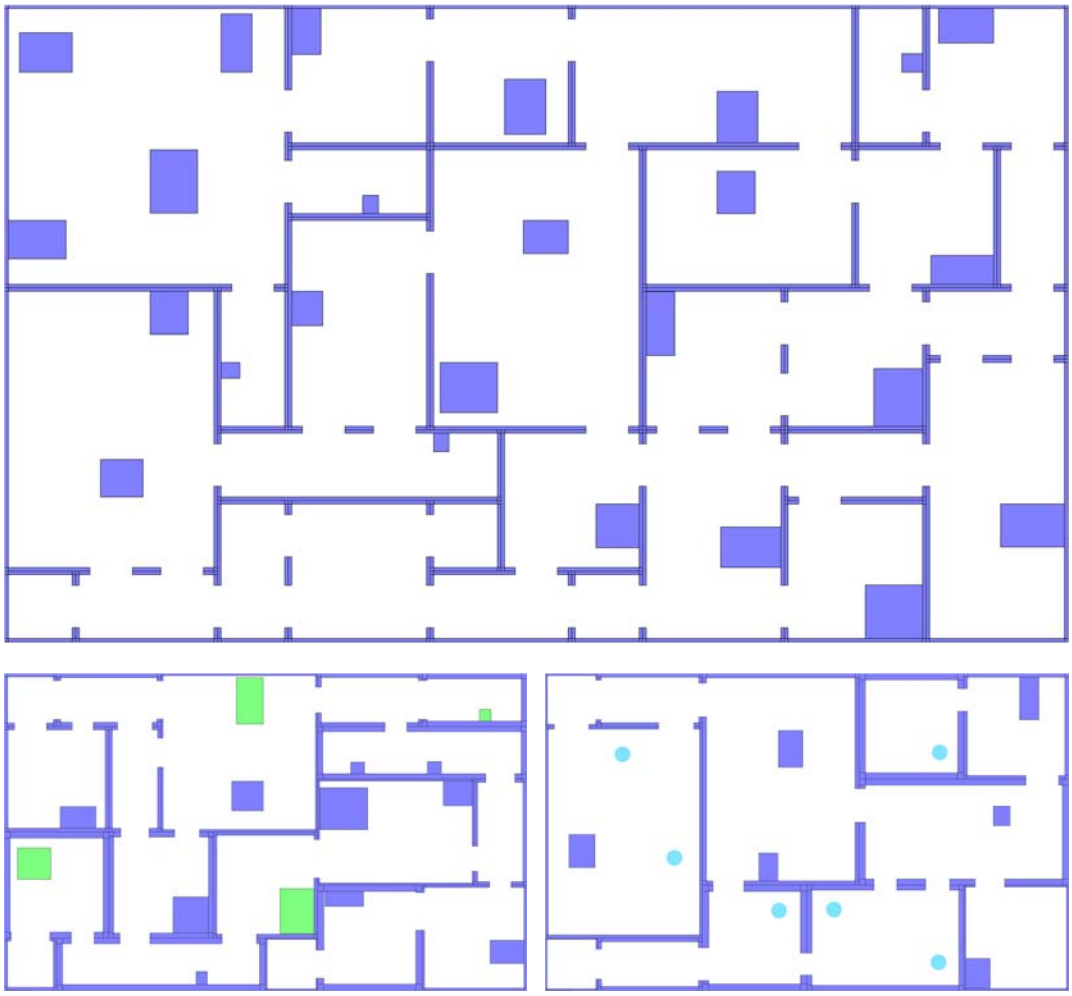

Figure 4.11: Example procedurally-generated environments.

The maximum number of doors allowed for a given room is proportional to its perimeter. Doors are only added to walls whose adjacent blocks are not yet occupied by other rooms. The order in which doors are placed is random. As each door is added, a new room of random dimensions is attached to it, and the process is repeated for the new room. If the new room cannot fit in the available space, the door connected to it is removed.

Following room placement, only one path exists from any one room to any other room, due to the restriction that each door only be added if a room has not yet been placed adjacent to it. While this also holds true for many small real-world environments, it does not provide an adequate challenge for path planning. Hence, a small number of additional doors are added to the environment, resulting in multiple paths between some rooms.

Sets of wall vertices are generated from the resulting structure. Their exact coordinates depend on pre-defined parameters such as wall and door widths.

Finally, objects are placed at random positions in each room. In order to prevent potential blockages, objects cannot be placed in blocks adjacent to doors or other objects. First, an object's type is randomly selected from the list of objects allowable for a particular experiment. Next, an object's size and other properties are determined within certain bounds:

- Static obstacles are generally rectangular, with randomly-generated positions, lengths and widths. In corridors, the maximum length for a static object is half the block width and it is always contained within a single block quadrant, to allow the robot sufficient space to navigate past it. In rooms, static objects can occupy up to a full block.
- Dynamic and moveable obstacles tend to be circular in order to simplify collision detection, and they are limited to less than half a block in width.

Figure 4.11 shows some examples of environments generated using this method.

4.7 Simulated Sensors and Actuators

The simulated version of MARVIN has one set of actuators (the driving motors) and three types of sensors (odometers, collision sensors and infrared distance measuring sensors).

4.7.1 Driving Motors

The simulated motor outputs are the robot's pose and polar velocities. First, the robot's target linear velocity v_T (in m/s) and angular velocity ω_T (in radians/s) are converted into wheel velocities (v_{T1}, v_{T2}), based on the distance between each wheel d_w (in m):

$$\begin{bmatrix} v_{T1} \\ v_{T2} \end{bmatrix} = v_T \pm \frac{d_w \omega_T}{2} \tag{4.1}$$

The simulated motors are driven by a simple PI controller. While a more complex controller may produce a more optimal response, it would likely be correspondingly more difficult to tune and less computationally efficient. Velocity errors v_E are

calculated each control cycle as the difference between the measured and target wheel velocities. Each wheel velocity v_W is a function of its velocity errors $v_E(k)$ at various integer times k, the proportional constant P and the integral constant I:

$$v_W(k+1) = v_W(k) + Pv_E(k) + I\big(v_E(k) + v_E(k-1) + v_E(k-2)\big) \tag{4.2}$$

Motor lag is modelled by averaging the current and previous wheel velocities. This is only an extremely crude approximation of physical motor properties, and no attempt was made to match those of MARVIN, but it is simple and efficient. Gaussian noise is also added to the motor output using MATLAB's *randn* random number generator function, whose output amplitude is modulated by velocity noise constant κ_v:

$$v_{W(noisy)} = \frac{(1 + \kappa_v \times randn)v_W(k+1) + v_W(k)}{2} \tag{4.3}$$

Static friction is controlled by a pair of constants $v_{\min 1}$ and $v_{\min 2}$. The magnitude of a wheel's velocity cannot not increase above zero until it exceeds $v_{\min 1}$. If a wheel's velocity magnitude drops below $v_{\min 2}$ from a non-zero number, it is set to zero.

Finally, the linear and angular velocities (v, ω) are obtained:

$$v = \frac{v_1 + v_2}{2} \tag{4.4}$$

$$\omega = \frac{v_1 - v_2}{d_w} \tag{4.5}$$

Pose values (x, y, θ) are a function of their previous values, the robot's polar velocities (v, ω), and its control period T:

$$\begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} x \\ y \end{bmatrix} + \frac{v}{\omega}\begin{bmatrix} \sin(-\theta) + \sin(\theta + T\omega) \\ \cos(-\theta) - \cos(\theta + T\omega) \end{bmatrix} \tag{4.6}$$

$$\theta \leftarrow \theta + T\omega \tag{4.7}$$

If the new pose results in a collision (see Section 4.7.3), the change is rejected, and the robot's linear velocity is set to zero. The previous pose values are selected in favour of the new ones, then shifted 1 cm away in the direction opposite to that of the point of collision.

4.7.2 Odometers

The outputs from the simulated odometers are the robot's internal representations of its pose and polar velocities. Due to odometer errors, the measured values can be different from the simulated robot's actual values.

Wheel velocities are again obtained from polar velocities using Equation 4.1. The measured wheel velocities (v_{M1}, v_{M2}) are a function of systematic error constants (ε_1, ε_2), and a random error constant ε_3 multiplied by the Gaussian random number generator function *randn*:

$$\begin{bmatrix} v_{M1} \\ v_{M2} \end{bmatrix} = \left(1 + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix} + \varepsilon_3 \begin{bmatrix} randn \\ randn \end{bmatrix} \right) \cdot \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{4.8}$$

Measured linear and angular velocities (v_M, ω_M) are obtained from the resulting measured wheel velocities using Equations 4.4 and 4.5. Similarly, the robot's measured pose values (x_M, y_M, θ_M) are obtained by applying Equations 4.6 and 4.7 to the measured polar velocities.

4.7.3 Collision Sensors

The simulated robot possesses eight collision sensors distributed evenly around its chassis, modelled as simple switches that trigger in the event of a collision. These are failsafe mechanisms that can help protect the robot from further damage if its other sensors and obstacle avoidance software prove inadequate. They are also used to record collisions during testing.

Objects in the robot's environment are decomposed into line segments representing their linked vertices. The collision detection algorithm determines whether these line segments occupy the same space as the robot.

First, for each line segment in the environment, a simple check is performed to determine if a circle enclosing it contains or intersects with a circle representing the robot:

$$\text{Cull line segment if} \left(x_L - x_R\right)^2 + \left(y_L - y_R\right)^2 > \left(r_R + \frac{l_L}{2} \right)^2 \tag{4.9}$$

This is a function of line segment's midpoint (x_L,y_L) and length l_L, and the robot's position (x_R,y_R) and radius r_R.

Next, if a line segment is not culled, the algorithm checks whether both of the line segment's endpoints are enclosed by the robot's circle. If not, it checks whether there are any points of intersection between the circle and line segment. A collision is simulated if either of these conditions is true. The point of collision is given by either

the line segment's midpoint, or the midpoint of the points of intersection, depending on which condition is true.

If any collisions are detected after all line segments in the environment have been processed, the sensors closest to the resulting points of collision are activated.

4.7.4 Infrared Distance Measuring Sensors

Both types of infrared sensor on MARVIN (Sharp GP2Y0A02YK and GP2Y3A003K0F) are modelled using the same algorithm. The only difference between them is the number of sensor beams modelled per sensor, and the ranges for each beam.

Each sensor beam is represented by a line segment extending from the emitter/detector's position, with a length bounded by its maximum sensing range. To detect objects in a beam's path, the algorithm calculates the points of intersection between the beam's line segment and each object's line segments. The measured object position is the closest point of intersection, if any exist.

The distance between the measured object position and the emitter/detector position represents the range d_{raw} that would be measured in the absence of sensor limitations. If d_{raw} is smaller than the sensor's minimum detection range, it is replaced by a new range value that is inversely proportional to d_{raw}.

$$d_{\text{raw}} \leftarrow d_{\min} + \left(d_{\min} - d_{\text{raw}}\right)\left(d_{\max} - d_{\min}\right) \tag{4.10}$$

Next, Gaussian noise is added to the range measurement d_{noisy} using *randn*:

$$d_{\text{noisy}} = d_{\text{raw}}\left(1 + \kappa_d \times randn\right) \tag{4.11}$$

The standard deviation of the noise distribution is proportional to the unmodified range value d_{raw} (by range noise constant κ_d). Although this model does not capture the full complexity of real-world sensor noise, it does approximate the effect where the noise intensity increases with range in real infrared sensors. This effect is linked to the non-linear shape of the voltage-distance relationship shown in Figures 4.4 and 4.5. The voltage variation between points at higher ranges is less than that at lower ranges, so noise has a larger influence over the resulting range measurement.

Due to the large amount of noise on both simulated and real measurements, a sliding median average filter is applied to the ranges before they are passed to the robot

controller. The filtered value is equal to the median of the five most recent samples. Thus, if the control period is set to its default 100 ms and the sensors are sampled once per control cycle, the average response time for the infrared distance measuring sensors is 250 ms. Figure 4.12 shows the resulting simulated IR sensor readings.

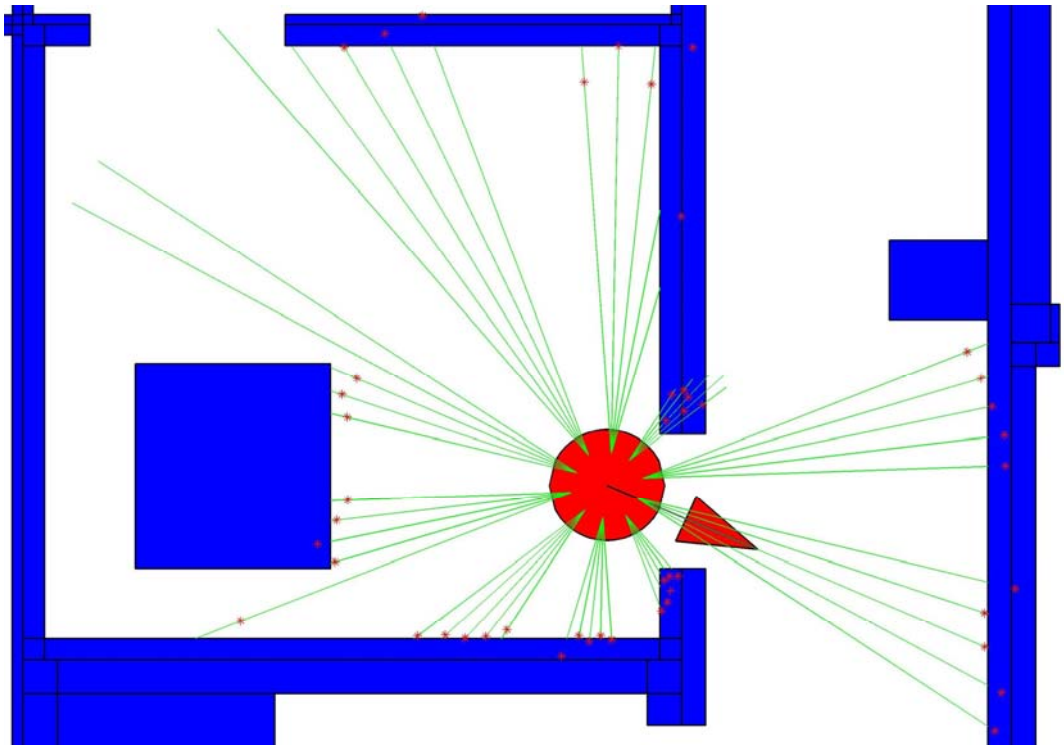

Figure 4.12: Simulated infrared distance measuring sensors for the robot in motion. The green lines represent the original noise-free, unfiltered ranges. The red asterisks represent the estimated obstacle positions after noise and median average filters are applied to the sensor readings.

4.8 Summary

The target platform for this research is MARVIN, a custom-built mobile security robot. MARVIN contains a wheeled base unit that enables the robot to navigate in flat-surfaced environments and a humanoid torso/head unit that facilitates human-robot interaction. A simulated mobile robot based on MARVIN has been developed for the purpose of testing control software in arbitrary 2D environments. These environments can be generated procedurally with a range of customisable properties.

The next chapter describes the implementation of the mobile robot planning and control system that is applied to this simulated version of MARVIN.

5 Motion Planning and Control Architecture

Mobile robot architectures can be broadly categorised by their reliance on representations of the environment. They generally reside somewhere along a spectrum between two main control methodologies: reactive and deliberative.

Reactive control methods advocate tightly coupled sensing and actuation, and a minimalist approach to world representation. Brooks (1986) argues that the world is its own best model, and complex behaviours can emerge from the interactions of simple components. Robots with reactive controllers tend to be fast, responsive to partially-observable dynamic environments, and tolerant of sensor noise. However, they are often unpredictable and prone to becoming trapped in local minima. Complex tasks requiring a robot to make long-term predictions or conform to rigid specifications generally cannot be accomplished with a purely reactive system.

Deliberative approaches involve planning a robot's actions in advance by searching a more global representation of the environment. Ideally, the robot's world representation is continually updated to incorporate new data, and its plans are adjusted accordingly. This allows a robot to solve certain problems that are beyond the reach of the trial-and-error approach employed by reactive systems. Computational overheads are unbounded, increasing exponentially as the quantity of world information increases. Thus, the design of a deliberative system always involves tradeoffs between speed and optimality. Purely deliberative robot controllers have traditionally been regarded as slow and unresponsive to environmental dynamics (Arkin, 1989).

Each of these approaches has significant limitations when utilised exclusively. Hence, many of the more successful implementations reside closer to the middle of the representation spectrum. Some are single-level systems that utilise partial environment representations, e.g. (Ulrich and Borenstein, 2000), while others are hybrid systems that combine multiple control methodologies, e.g. (Gat, 1992), (Conell, 1992).

A hybrid reactive/deliberative control approach is employed in this thesis. Hybrid controllers have been increasingly popular in recent years, as they can mitigate many of the weaknesses of single-level systems. One of the key issues of hybrid controller

design is management of the interactions between reactive and deliberative components. In our architecture, the deliberative controller is assigned a loose supervisory role, providing recommendations that can be obeyed or overruled by the reactive controller as the situation dictates. This allows the robot to benefit from the guidance of deliberative planning, while maintaining the real-time responsiveness of reactive control. An overview of the controller is given in Figure 5.1.

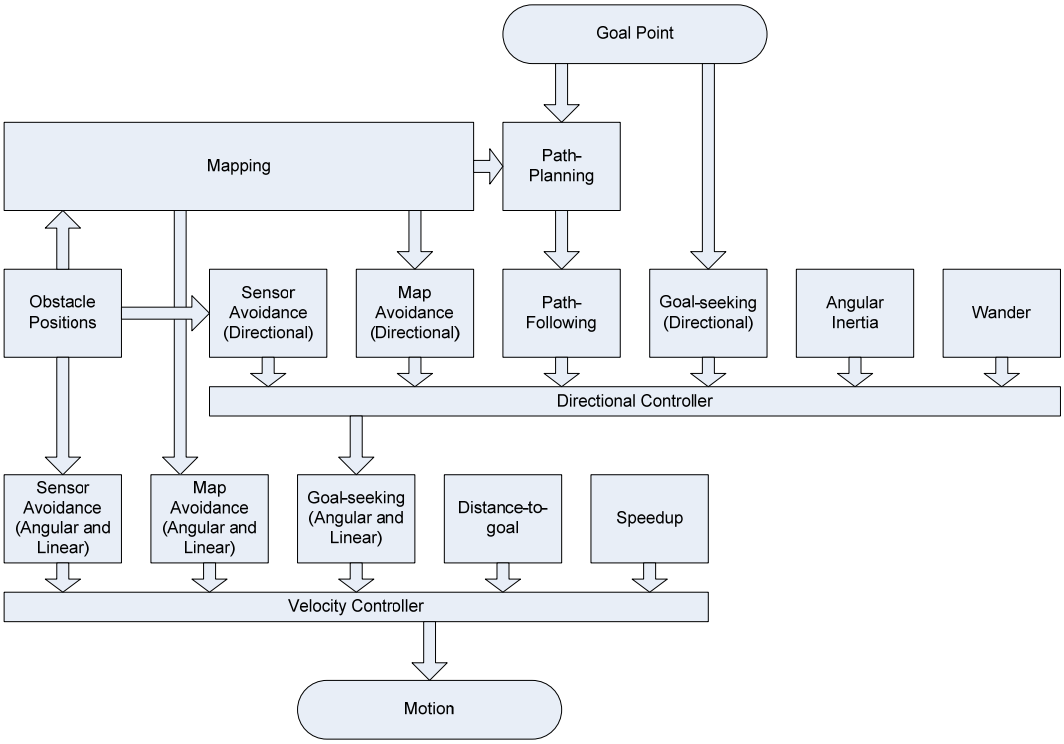

Figure 5.1: A block diagram representation of the hybrid reactive/deliberative navigation system.

In accordance with authors such as Brooks (1986) and Gat (1992), this architecture was designed largely from the bottom up, starting with a purely reactive system, then incorporating deliberative planning capabilities. However, subsequent sections describe the architecture components in a top-down manner. This is because recent additions to the reactive layers are dependent on deliberative world representations. For example, the map-based obstacle avoidance functions utilise global map data.

5.1 Deliberative Control

Reactive obstacle avoidance techniques are susceptible to becoming trapped in local minima. This problem is often alleviated by injecting noise into the system, or by

detecting and responding appropriately to local minima when they are encountered. However, convergence to the goal in a reasonable timeframe can only be guaranteed if they are combined with some form of global path planning. Our reactive controller is no exception, and its reliability can be improved by the addition of a deliberative planner.

Two main components form the deliberative controller: dynamic mapping and path planning. Mapping is performed continuously as the robot explores its environment, whereas paths are planned less frequently on a separate timer.

The problem of localisation is considered 'solved' for the purposes of this research, so the methods described in this section assume that the robot has accurate knowledge of its position in the environment. This is not entirely realistic, as existing simultaneous localisation and mapping (SLAM[1]) approaches have significant limitations. For example, when entering a room from two different doorways, many SLAM algorithms fail to identify it as the same room each time. Nevertheless, the SLAM problem is beyond the scope of this project.

5.1.1 Mapping

Path planning typically involves the application of a search algorithm or heuristic to a graph structure representing a set of valid paths obtained from a map of the robot's environment. There are two main categories of path planning methods, each tied to different types of map representation.

In cell-decomposition methods, obstacles and free space are divided into discrete cells (Hwang and Ahuja, 1992). Valid paths are obtained from the adjacency relationships between free cells. Selection of cell sizes is generally a trade-off between path quality and computational efficiency. Large environments can result in unacceptably large search spaces unless they are partitioned into manageable segments.

Roadmap methods such as visibility graphs (Nilsson, 1969), Voronoi diagrams (Aurenhammer, 1991), freeway nets (Latombe, 1991) and silhouette graphs (Canny, 1988) generate a set of curves in free space that connect nodes between obstacles. These approaches can greatly reduce the search space without impeding path quality, but the overhead of generating the roadmaps can negate any computational efficiency

[1] Durrant-Whyte and Bailey (2006) give a concise review of SLAM approaches.

gains, and they are often best suited to specific types of environments. For example, Voronoi diagrams perform well in confined indoor environments, but poorly in open outdoor environments.

5.1.1.1 Occupancy Grid Map

Our robot's environment is represented by a rectangular occupancy grid map (Thrun, 2003). This is selected over roadmap methods and other types of cell decompositions because of its simplicity and versatility. An example map is shown in Figure 5.2.

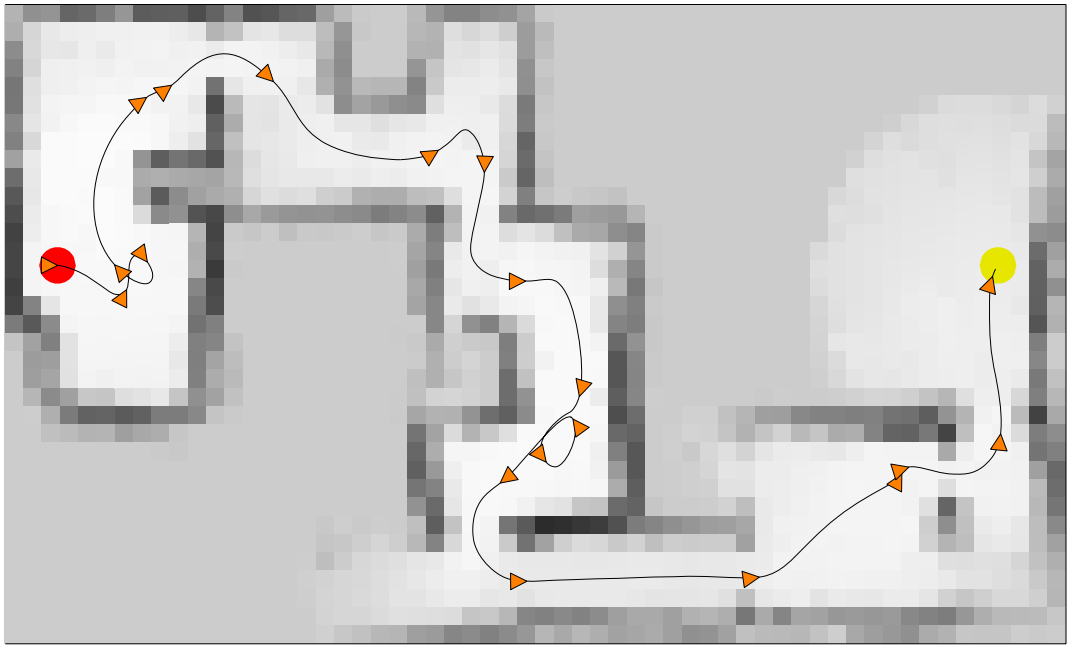

Figure 5.2: An occupancy grid map generated during one pass through a simulated environment. Darker shades represent increased probability of occupancy. Orange direction arrows are placed at 5 second intervals along the path; sparse arrow densities represent higher speeds. Start and goal points are represented by the red and yellow circles, respectively.

The grid size is set to the robot's radius (35 cm). This results in a coarse grid resolution that facilitates computationally efficient mapping and path planning, but nevertheless provides sufficient data for planning purposes. Obstacle avoidance and local navigation is largely unimpeded, because it is more dependent on local sensor data than on global maps (although both can be taken into consideration). Hence, any minor benefits to be gained by increasing the grid resolution are likely not worth the cost to computational efficiency.

Each node is assigned an occupancy probability p_o, a unit interval value that represents the estimated probability that the node is occupied by an obstacle. Occupancy probabilities are updated in real time based on proximities to measured obstacles and sensor beams.

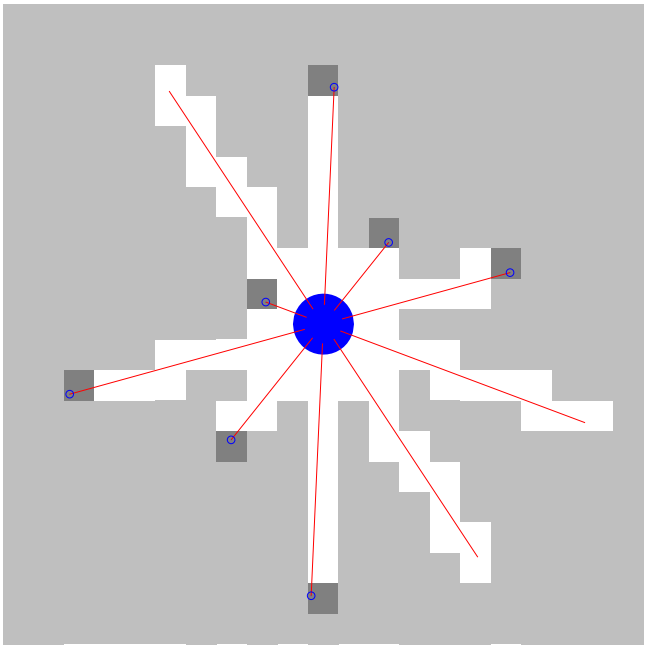

Figure 5.3: Obtaining occupancy status from sensor data. The large blue circle represents the robot. Sensor beams are represented by red lines, and obstacle positions are marked by small blue circles. Map nodes flagged as occupied are dark grey, while unoccupied nodes are white.

Each sensor is represented by a line that extends from its origin to the obstacle detected, bounded by a predefined maximum sensor range. Nodes through which a sensor beam passes are flagged as unoccupied. Detection of such nodes involves calculating the points of intersection between relevant line segments. Several optimisations are employed to ensure that only potentially-relevant nodes are processed, and that the line segments processed are shared by multiple nodes. Nodes within radius r of the robot's current position (i.e. nodes currently occupied by the robot) are also flagged as unoccupied by obstacles. Nodes containing detected obstacles are flagged as occupied, superseding any unoccupied status resulting from proximity to the robot or its sensor beams. A diagram of this process is shown in Figure 5.3 (although the actual robot produces more sensor beams than are shown in the diagram).

The occupancy status of each flagged node is an instantaneous value (unoccupied = 0, occupied = 1) that is incorporated into the node's occupancy probability calculation. This calculation is a weighted average of the occupancy status $s_o(x)$ and the previous occupancy probability $p_o(x)$ of each node x:

$$p_o(x) \leftarrow \varepsilon s_o(x) + (1-\varepsilon) p_o(x) \tag{5.1}$$

The map update weight ε controls the rate at which detected occupancy changes are incorporated into the map, balancing the competing demands of speed and stability. High values of ε result in faster response to detected obstacles, but increased instability. Separate update weights can be employed for increasing and decreasing obstacle probabilities.

5.1.1.2 Other Grid Maps

Occupancy probability is not the only variable associated with map nodes. Other variables such as exploration, danger and emotions are mapped in order to provide location-specific biases to the robot's planning and control systems.

An exploration map is constructed simultaneously with the occupancy map. The purpose of this map is to provide the robot with an incentive to explore regions that are not currently known. Each node x is assigned a value $p_e(x)$ representing a degree of confidence that the node is fully known. Updating $p_e(x)$ is accomplished in the same manner as for $p_o(x)$, but growth is applied uniformly to all nodes whose $p_o(x)$ values are updated, rather than being a function of $s_o(x)$:

$$p_e(x) \leftarrow \varepsilon + (1-\varepsilon) p_e(x) \tag{5.2}$$

Thus, $p_e(x)$ values do not decay in our current implementation. Once explored, a node will always be explored. However, in environments where a robot's existing world knowledge may become obsolete, a time dependent decay factor could be introduced. Figure 5.4 shows an exploration map acquired simultaneously with the previous occupancy map (Figure 5.2).

A danger map is employed to mark regions that the robot should attempt to avoid. These include regions where there are fast-moving obstacles (e.g. humans), objects that the robot's sensors cannot easily detect (e.g. windows), or objects that result in severe consequences if the robot should fail to avoid them (e.g. staircases). Map nodes can be tagged as dangerous either a priori, or by recording points of collision as they

occur. Danger is currently represented as binary (dangerous/safe) states, but it could be modified to represent more subtle degrees of danger. The robot is discouraged from planning paths through these regions, and these nodes also influence reactive obstacle avoidance in certain situations.

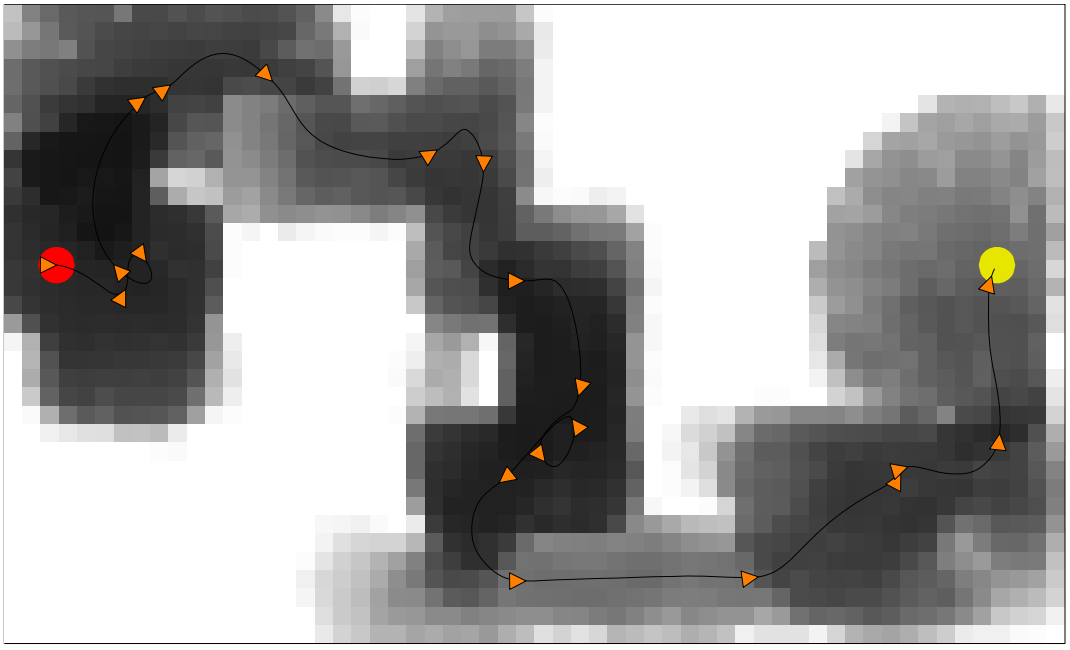

Figure 5.4: An exploration grid map. Darker shades represent increased confidence of knowledge.

Emotions are also associated with map nodes, to provide positive or negative location-specific biases. These are described further in later chapters. Other types of maps can be incorporated as required by a specific application. For example, regions of strategic importance to the robot such as patrol routes or off-limit zones could be marked to increase or decrease their attractiveness.

5.1.1.3 Map Fusion

Grid maps are fused by weighted probabilistic OR calculations. For all nodes x, two input map values m_1 and m_2 are combined into a fused output map value m_f:

$$m_f(x) = W_1 m_1(x) + W_2 m_2(x) - W_1 W_2 m_1(x) m_2(x) \tag{5.3}$$

The fused map m_f, not the occupancy map, is utilised during path planning. Weights W_1 and W_2 are unit interval values representing the relative contributions of each map. The occupancy map is typically assigned a weight of 1, but the weights of other maps vary depending on the task at hand. For example, if the exploration map is highly

weighted, the robot tends to favour exploration over finding the shortest path. A combined occupancy and exploration map is shown in Figure 5.5 (although exploration was assigned zero weighting during the acquisition of its component maps). To fuse more than two maps (m_3-m_n), this calculation can be repeated as many times as are necessary:

$$\begin{aligned} m_f(x) &\leftarrow m_f(x) + W_3 m_3(x) - W_3 m_f(x) m_3(x) \\ &\ldots \\ m_f(x) &\leftarrow m_f(x) + W_n m_n(x) - W_n m_f(x) m_n(x) \end{aligned} \tag{5.4}$$

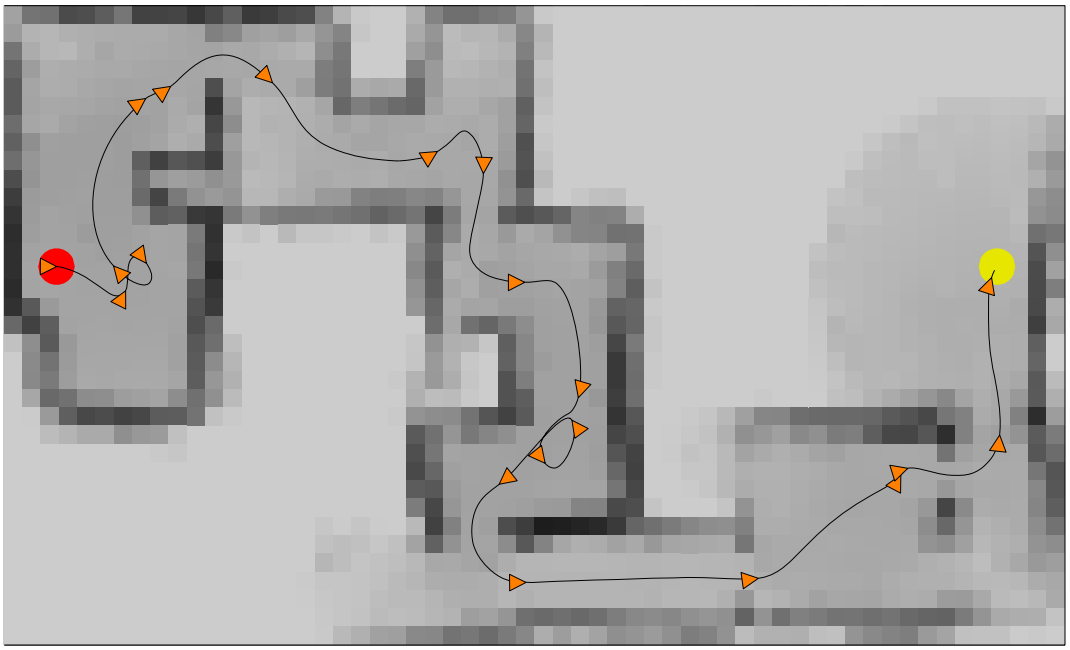

Figure 5.5: Occupancy and exploration grid maps fused into a single map. Variables: W_1 = 1, W_2 = 0.4.

5.1.2 Path Planning

The path planner utilises the A* algorithm (Judea, 1984) to find an optimal path through the fused grid map. A* is a best-first graph search algorithm that prioritises nodes by the estimated quality of their associated paths. A pseudo-code implementation of the A* algorithm is shown in Figure 5.6.

The total path cost $f(x)$ of a node x is dependant on a measured path cost $g(x)$ of the best path from the start node to x, as well as a heuristic path cost $h(x)$ of travel from x to the goal node:

$$f(x) = g(x) + h(x) \tag{5.5}$$

```
Add start node to open list.
Repeat until goal node is on closed list:
        If open list is empty, return failure.
        Set parent node to node in open list with minimum total path cost.
        Move parent node from open list to closed list.
        For each child node adjacent to parent node:
                If child node is not on either open list or closed list:
                        Calculate total path cost of child node.
                        Move child node to open list.
Set selected node to goal node.
Add selected node to path list.
Repeat until selected node is start node:
        Set selected node to parent of selected node.
        Add selected node to path list.
Reverse path list.
```

Figure 5.6: Pseudo-code representation of the A* implementation.

Measured path cost $g(x)$ is dependant on the cost of the lowest-cost parent node x_{par}, the Euclidian distance $d_n(x, x_{par})$ between the two nodes, and a cost factor $c(x)$:

$$g(x) = g(x_{par}) + d_n(x, x_{par})c(x) \tag{5.6}$$

In our implementation, $c(x)$ is a continuous value dependant on a filtered map node cost $m_f'(x)$:

$$c(x) = B^{m_f'(x)} \tag{5.7}$$

Base constant B is an arbitrarily large number (typically $B > 10^3$) that affects the disparity between cost factors of nodes with different m_f' values.

This contrasts with standard A* path planning methods, where nodes x whose occupancy values exceed a threshold are regarded as occupied by obstacles, and are therefore not processed during path planning. Although they do not include an explicit variable equivalent to $c(x)$, the approach is analogous to assigning $c(x) = \infty$ to nodes that exceed the threshold, and $c(x) = 1$ to all other nodes, regardless of their $m_f(x)$ values. A visual comparison between the two approaches is shown in Figure 5.7.

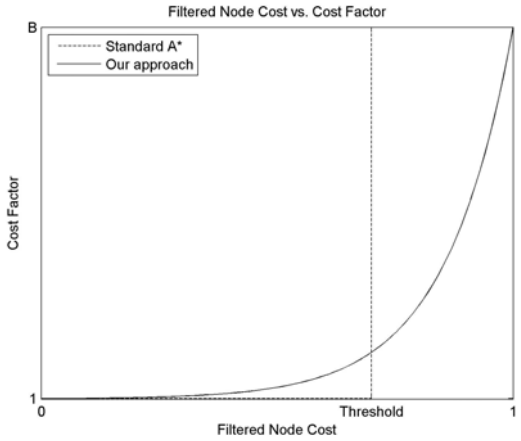

Figure 5.7: The relationship between $m_f'(x)$ and $c(x)$ in our approach, and its analogue in standard A* path planning methods.

Our approach allows any changes in map values to influence path planning, not just those that cross a threshold. This is important to our application because path planning is not just a function of occupancy probability, but also other factors such as exploration and emotions. The subtleties introduced by these other influences would be negated if binary thresholds were employed. Due to the exponential nature of the relationship, the robot is unlikely to select a node with a high cost when low-cost alternatives exist. However if no unobstructed paths exist, the planning algorithm fails gracefully by choosing the best of the unfavourable options available. The reactive controller prevents the robot from colliding with obstacles even if it is instructed to pass through them, and it can sometimes reactively navigate through nodes that the planner regards as occupied.

A smoothing filter is employed to obtain $m_f'(x)$ for a given node x from the fused map m_f:

$$m_f'(x) = \max \begin{bmatrix} \left(1 - \frac{d(x_1)^2}{r_f^{\,2}}\right) m_f(x_1) \\ \left(1 - \frac{d(x_2)^2}{r_f^{\,2}}\right) m_f(x_2) \\ \cdots \\ \left(1 - \frac{d(x_n)^2}{r_f^{\,2}}\right) m_f(x_n) \end{bmatrix} \tag{5.8}$$

This filter is applied to nodes x_1-x_n within filter radius r_f of node x. The contribution of each node is weighted by a function of its Euclidian distance d from node x. The

advantage of this approach over a simpler low-pass filter is that it smoothes the edges of low-cost regions without reducing the node costs of high-cost regions. This encourages the robot to plan paths closer to the centre of free space regions, which generally results in smoother paths that only pass near obstacles if no other option is available.

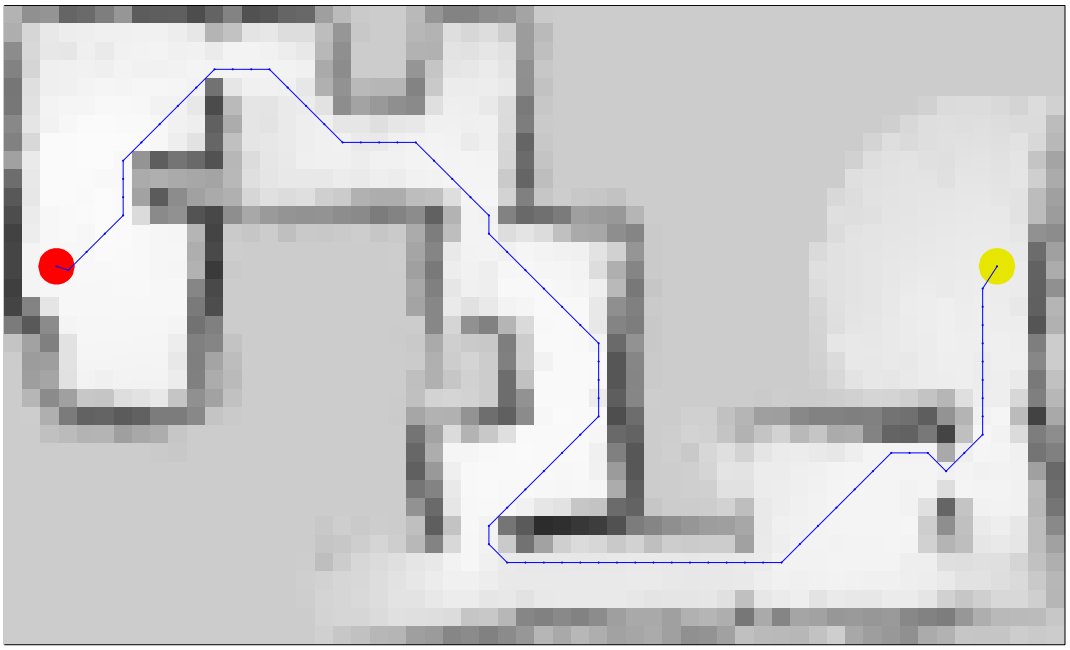

Figure 5.8: A path planned without map filtering.

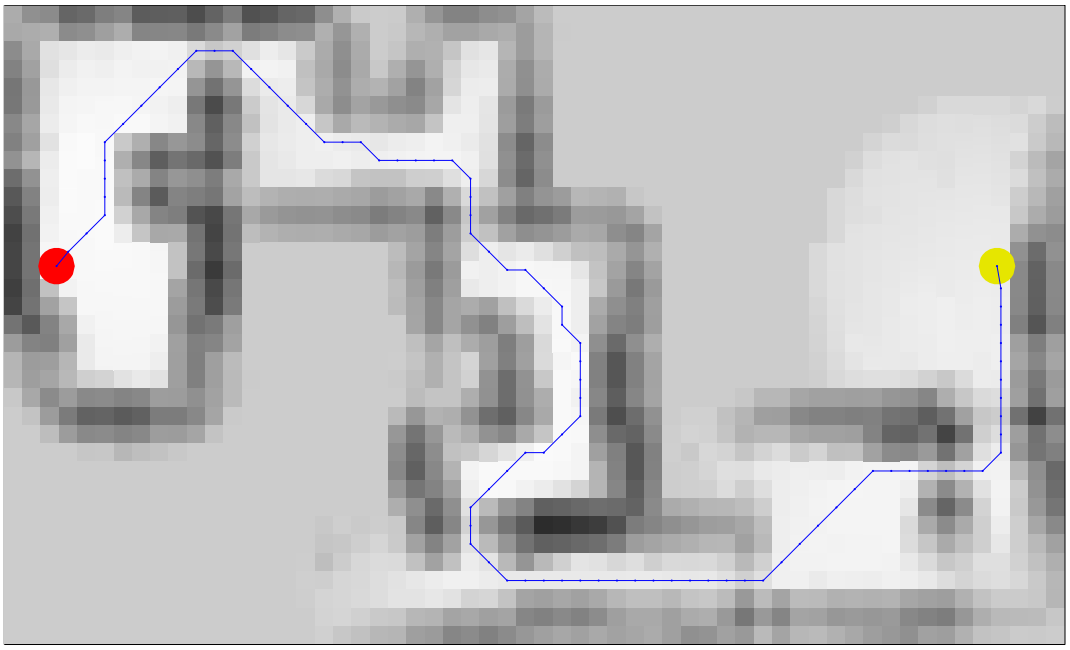

Figure 5.9: A path planned with filter radius $r_f = 0.7$.

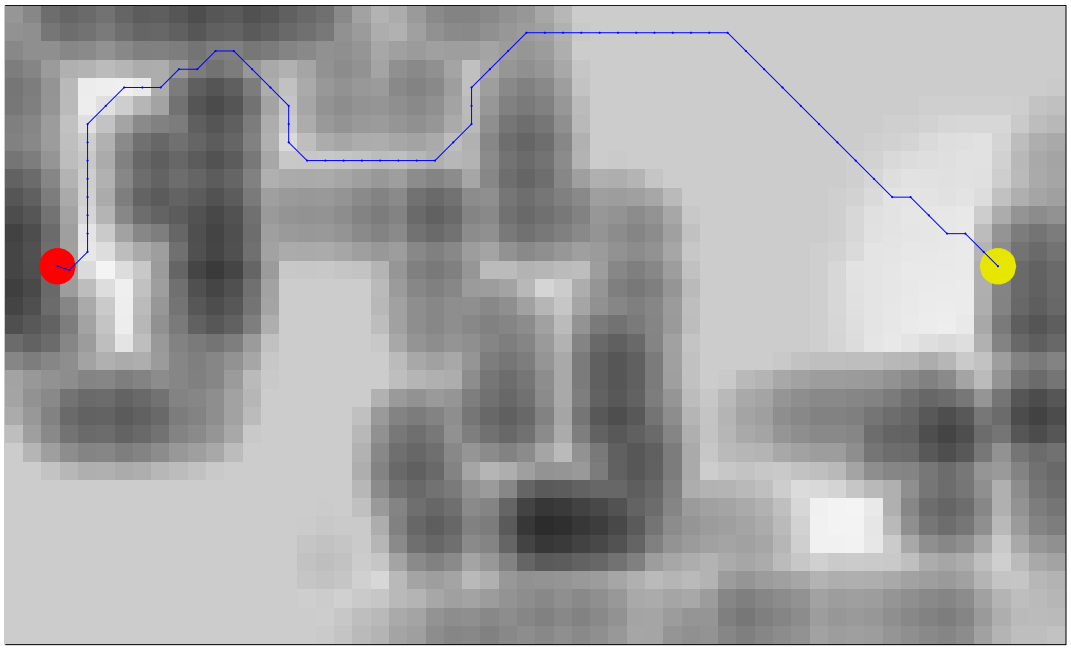

Figure 5.10: A path planned with filter radius $r_f = 1.5$.

The filter radius r_f influences the degree of smoothing, which can directly affect the quality of the planned path, as shown in Figure 5.8, Figure 5.9 and Figure 5.10. The path shown in Figure 5.8 is a result of unfiltered planning, and it tends to approach close to the edges of walls. If a moderate level of filtering is applied, as shown in Figure 5.9, the path is relatively optimal. If the map is over-filtered, as shown in Figure 5.10, doorways tend to be filtered, resulting in a sub-optimal path. Although the smoothing is shown applied to the whole map, the robot actually only applies it to nodes that are processed by the path planning algorithm, for computational efficiency reasons.

Heuristic cost $h(x)$ is the Euclidian distance between x and the goal node. This is an optimistic estimate; the actual cost is generally larger than the straight-line distance, especially once the additional costs of non-zero m_f' values are taken into consideration. As a result, the algorithm is heavily biased towards optimality rather than computational efficiency. This approach is acceptable for our implementation, because the limited grid resolution and environment sizes result in small search spaces.

A new path is planned not only upon the receipt of a new instruction, but also periodically throughout the journey. This allows the robot to take advantage of any world knowledge acquired since the previous plan. Planning is triggered whenever a

timer exceeds the replan period t_r. Lower values of t_r improve the robot's response to detected environmental changes, whereas higher values reduce the computational resources allocated to planning.

5.2 Reactive Control

Despite the existence of a deliberative path planner, our reactive controller is designed to be self-sufficient. In most situations it does not require a path planner to function, albeit at a reduced capacity. This is important because deliberative systems cannot always be relied upon to provide useful commands. Environments that are highly dynamic or sparsely occupied tend to negate the advantages of planning. In the former case, the robot's plans may be obsolete by the time they are executed; in the latter, there are few local minima to obstruct a reactive controller. Furthermore, deliberative systems are strongly dependent on localisation accuracy. A global map is only useful if a robot 'knows' its location. Hence, localisation errors are less likely to result in critical failures if the reactive system can operate independently when required.

The controller must manage competing objectives such as goal seeking, obstacle avoidance, speed and stability. The relative importance of these objectives may change depending on the robot's task and environment. Hence, one of our primary considerations is flexibility. Whereas many controllers are heavily reliant on thresholds to constrain decisions for computational efficiency reasons, this controller usually favours continuous values over binary decisions. A number of weights and parameters can be adjusted dynamically depending on the intensities of the robot's affective states and processes. The affect model is described further in subsequent chapters.

The controller is divided into two stages. First, it seeks an appropriate direction that approaches the goal location or planned path while attempting to avoid obstacles, oscillations and local minima. Second, it finds a pair of velocities that move the robot in the target direction at an appropriate speed while avoiding obstacles.

5.2.1 Directional Control

One of the main inspirations for our directional control method is the vector field histogram (VFH) obstacle avoidance technique (Borenstein and Koren, 1991). In its

original implementation, VFH is applied to a probabilistic occupancy grid map constructed from sonar sensor data. Direction space is discretised into a polar histogram of candidate sectors. Each sector is assigned a polar obstacle density derived from the certainty values of the cells it encompasses. The obstacle densities are smoothed, and a threshold is employed to sort the sectors into peaks and valleys. The robot attempts to travel through the centre of the valley closest to the direction of the target point.

The VFH+ method (Ulrich and Borenstein, 1998) improved on VFH in a number of ways. Obstacles are padded by a circle of the robot's radius, explicitly preventing the robot from attempting to pass through openings that are too narrow. Hysteresis is applied to the sorting threshold, preventing oscillations as the robot passes regions close to the threshold. Sectors that are blocked by obstacles in other sectors are masked, taking into account that wheeled robots tend to move in circular arcs rather than straight lines. A target direction is selected for the robot by applying a weighted sum objective function to each valley. In addition to the direction of the target point, the objective function takes into account the current wheel orientation and the previously-selected target direction.

5.2.1.1 Objective Function

Like VFH and its subsequent refinements, the method utilised by our controller involves the construction of a polar histogram, but many other implementation details are markedly different. Rather than separating obstacle avoidance from other objectives such as goal seeking, all objectives are integrated into a single weighted product objective function. Given a goal location and/or planned path, the directional controller obtains a locally-optimal target heading θ_T by applying the objective function $f(\theta)$ to a discrete list of candidate headings θ:

$$\theta_T = \arg\max_{\theta \in [-\pi, \pi)} f(\theta) \tag{5.9}$$

The objective function f is a combination of individual objectives f_1–f_n modulated by weights W_1–W_n (where f_1–f_n and W_1–W_n are unit interval variables):

$$f = \left(1 - W_1\left(1 - f_1\right)\right) \times \left(1 - W_2\left(1 - f_2\right)\right) \times \ldots \times \left(1 - W_n\left(1 - f_n\right)\right) \tag{5.10}$$

This form has some advantages over the weighted sum utilised by methods such as VFH+ (and various velocity-space approaches discussed in Section 5.2.2):

$$f = \frac{W_1 f_1 + W_2 f_2 + \ldots + W_n f_n}{W_1 + W_2 + \ldots + W_n} \tag{5.11}$$

These advantages are revealed in Figure 5.11 and Figure 5.12, which show the outputs of two-input objective functions of each form. Input f_1 represents an important objective, while input f_2 represents an objective of lower importance.

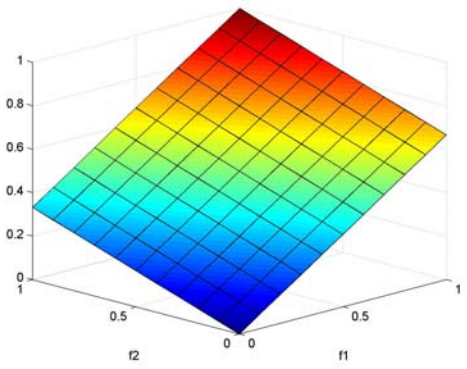

Figure 5.11: Weighted sum objective function with $W_1 = 1$, $W_2 = 0.5$.

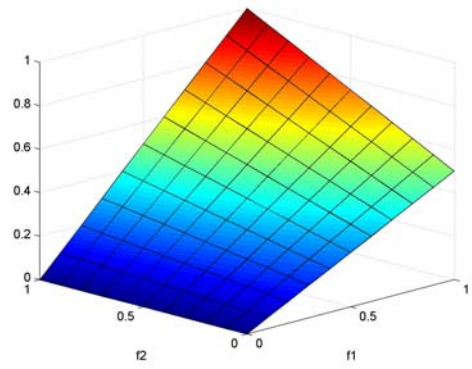

Figure 5.12: Weighted product objective function with $W_1 = 1$, $W_2 = 0.5$.

The weighted sum objective function (Equation 5.11, Figure 5.11) allows less important objectives to influence the output even when a more important objective indicates that a particular decision is highly unfavourable. This can result in the robot choosing dangerous actions under certain circumstances. For example, the function f_1 shown in Figure 5.11 might represent obstacle avoidance, while f_2 could represent goal seeking. In that scenario, a 'dangerous' option of $f_1 = 0$ and $f_2 = 1$ produces a higher objective value ($f = 0.33$) than a 'safe' option of $f_1 = 0.4$ and $f_2 = 0$ ($f = 0.27$). Thus, the 'dangerous' option is favoured by the controller, even though it is significantly more likely to result in a collision than the 'safe' option.

In contrast, the weighted product objective function (Equation 5.10, Figure 5.12) employed by our controller enables an important objective to negate the possibility of the robot choosing a highly unfavourable option (as long as other more favourable options exist), while still allowing less important objectives to exert a high level of influence over options it regards as moderately unfavourable. In the scenario described above, the 'dangerous' option would yield $f = 0$, while the 'safe' option would result in $f = 0.2$ and would therefore be selected. A large number of low-weighted objectives can be incorporated into the system without adversely affecting the contributions of the most important objectives. The weights are typically limited

to 0.99 or less (i.e. close to 1, but not equal to 1), so that if a highly important objective determines that all available options are completely unacceptable, the other objectives remain able to contribute to the robot's decisions.

5.2.1.2 Avoidance Function

The most important component of the objective function is the avoidance function $a(\theta)$, which favours directions with more distant obstacles. Whereas VFH utilises obstacle density values derived from an occupancy grid, our controller rates each polar sector by the distance to its closest obstacle obtained from absolute obstacle coordinates. This method does not take sensor uncertainties into account. However, it allows the robot to respond quickly to obstacles. Unlike VFH, sectors are not assigned binary peak/valley status. Instead, obstacle distances are converted into continuous values representing estimated safety.

Infrared sensor ranges and any measured points of collision are converted into absolute obstacle positions, which are stored in a buffer containing the previous n obstacle measurements. The value of n can be adjusted to change the bias between computational efficiency and control performance.

The robot is roughly circular, so for simplicity it is represented as a point object, and each obstacle is represented by a circle of radius r_o. Typically, r_o is equal to the robot's radius, but lower values may allow the robot to enter narrow doorways, and higher values may improve safety. The points of intersection between each circular obstacle and a line extending out from the robot's position in direction θ are calculated (Figure 5.13).

If this method is applied to a robot whose chassis cannot be approximated by a circle, the circular obstacles can be replaced by polygonal objects constructed from a base shape rotated to match the robot's current orientation. The radius r_o can be replaced by a scale factor employed to grow or shrink the obstacles.

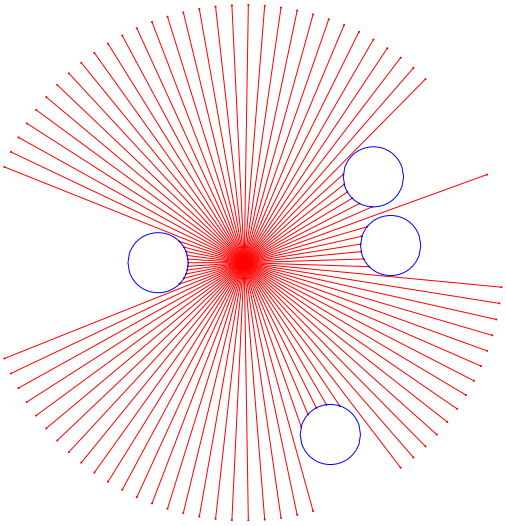

Figure 5.13: Unfiltered avoidance vector field (red lines) with circular obstacles (blue circles).

The value of $a(\theta)$ is calculated from the distance $d_o(\theta)$ between the robot's position and the closest point of intersection, normalised by the maximum sensing range $d_{o(\max)}$:

$$a(\theta) = \begin{cases} \dfrac{d_o(\theta)}{d_{o(\max)}} & \text{if } d_o(\theta) < d_{o(\max)} \\ 1 & \text{otherwise} \end{cases} \tag{5.12}$$

The resulting values are filtered over all angles θ to reduce the attractiveness of directions that pass near obstacles, compelling the robot to give obstacles a wide berth and favour the centre of corridors and doorways. The original VFH method utilised a simple weighted moving average low-pass filter. VFH+ enlarges obstacles by the robot's radius as a replacement for this filter. However, as shown in Figure 5.13, enlargement alone does not prioritise between directions that pass extremely close to an obstacle and those that are further away. In this example, one vector that passes between two adjacent obstacles is assigned equal priority to those that are nowhere near obstacles.

Hence, both filtering and enlargement are employed in our controller. The low-pass filter utilised by VFH has the disadvantage that it smoothes both valleys and peaks. This may occasionally provide the robot with insufficient incentive to avoid brushing

the corners of obstacles. A superior approach for our implementation is to utilise a filter that only smoothes the edges of peaks, not valleys[2].

Given a vector field h with n vectors, the filter applied to each element k is given by:

$$h'(k) = \min \begin{bmatrix} h(k-m) + \dfrac{m}{sn} \\ \cdots \\ h(k-1) + \dfrac{1}{sn} \\ h(k) \\ h(k+1) + \dfrac{1}{sn} \\ \cdots \\ h(k+m) + \dfrac{m}{sn} \end{bmatrix} \tag{5.13}$$

The number of contributing elements m in each direction from element k is a (rounded down) function of smoothing factor s and the total number of vectors n:

$$m = floor(sn) \tag{5.14}$$

Figure 5.14 and Figure 5.15 show vector fields resulting from smoothing factors of 0.1 (which smoothes up to 10% of the vector field in each direction) and 0.2, respectively. Increasing s encourages the robot to approach the centres of peaks, resulting in improved safety. However, large values (e.g. Figure 5.15) may discourage the robot from traversing narrow doorways or other 'bottleneck' regions.

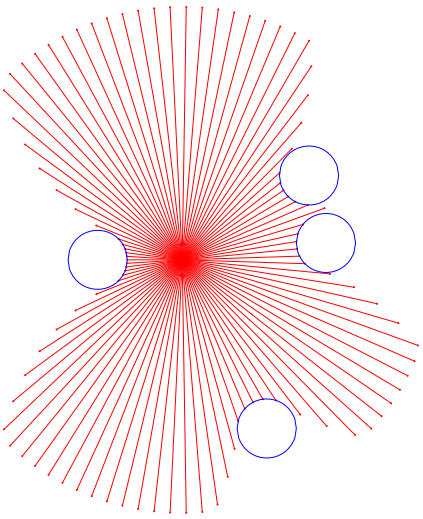

Figure 5.14: Filtered avoidance vector field with s = 0.1.

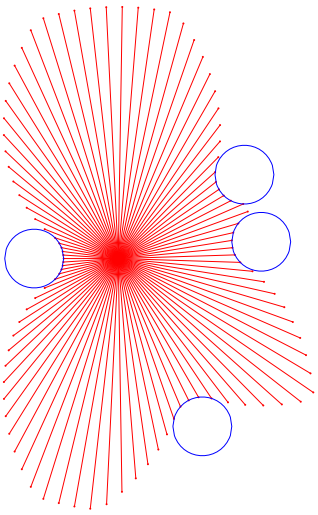

Figure 5.15: Filtered avoidance vector field with s = 0.2.

[2] Our representation of peaks and valleys is opposite to VFH, because we measure obstacle distance, not obstacle density.

5.2.1.3 Map-based Avoidance Function

The sensor-based avoidance function enables the robot to avoid obstacles detected by its sensors, but certain objects may be transparent or may reside outside of its plane of detection, essentially rendering them 'invisible' to the robot. To help address this problem, a map-based avoidance function is incorporated into the system. Most aspects of this objective are the same as its sensor-based equivalent, but instead of obtaining obstacle positions from sensor readings, it constructs virtual obstacles from the robot's internal maps.

Of the available grid maps, only the occupancy map and danger map are utilised by this function. They are combined using Equation 5.3 to obtain a fused map m_f. Nodes x within a radius r of the robot's current position whose $m_f(x)$ values exceed a threshold p_T are regarded as virtual obstacles. A virtual obstacle's position is given by the centre coordinates of its map node. Once a list of virtual obstacles is obtained, the vector field is constructed in an identical manner to the sensor-based avoidance function.

5.2.1.4 Goal Seeking Function

If it only had avoidance functions, the robot would wander aimlessly through free space. Additional objectives are necessary to provide goal-directed behaviour. One of these is the goal seeking function $g(\theta)$, which gives preference to directions θ that are closer to the direction θ_g of the goal point (Figure 5.16). The value of $g(\theta)$ is given by:

$$g(\theta) = 1 - \frac{\left| adjust\left(\theta - \theta_g\right) \right|}{\pi} \tag{5.15}$$

The *adjust* function ensures that angles always reside within the interval $[-\pi, \pi)$.

Avoidance and goal seeking objectives, once combined into a single objective function using Equation 5.10, are sufficient to produce simple reactive navigation in sparse environments. Figure 5.17 shows a vector field combining those shown in Figure 5.14 and Figure 5.16. In this example, the vectors pointing southeast are clearly favoured over all others, due to the absence of obstacles in the goal direction, so that would be the direction selected by the robot.

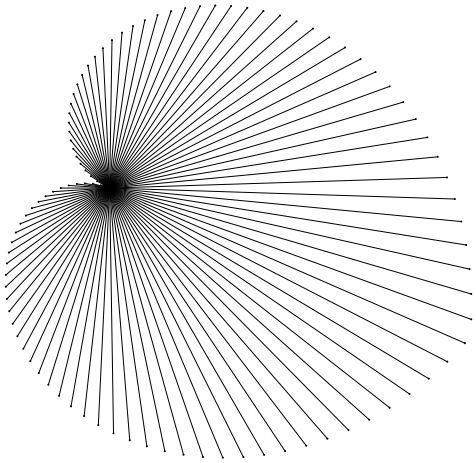

Figure 5.16: Goal seeking vector field – vector size increases with proximity to target angle.

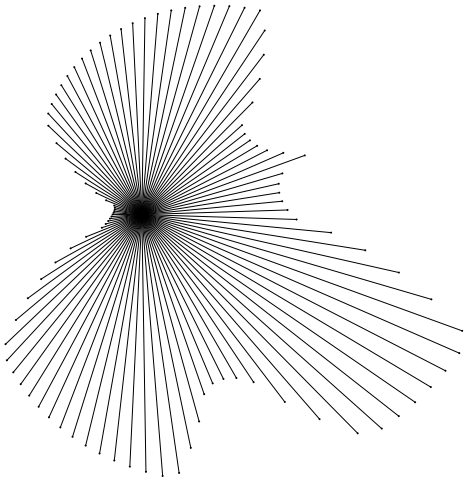

Figure 5.17: Combined avoidance and goal seeking vector field. Variables: avoidance weight $W_1 = 1$, goal seeking weight $W_2 = 0.7$, $s = 0.1$.

5.2.1.5 Angular Inertia Function

Additional objectives are required to facilitate reactive navigation in cluttered environments. One of these is the angular inertia function $i(\theta)$, which favours smaller changes in direction. This prevents the robot from oscillating between multiple directions that are otherwise equally favourable. It also allows the robot to escape from some local minima by increasing its 'persistence' in attempting to move around obstacles. The value of $i(\theta)$ is dependent on the previous target direction $\theta_{T(prev)}$:

$$i(\theta) = 1 - \frac{\left| adjust\left(\theta - \theta_{T(prev)}\right) \right|}{\pi} \tag{5.16}$$

5.2.1.6 Wander Function

The combination of avoidance, goal seeking and angular inertia objectives allow the controller to function similarly to other directional controllers such as VFH. The robot can avoid most static obstacles, and it can find its way to the goal location in many environments. Nevertheless, it is prone to becoming trapped in local minima. This basic functionality is extended by the wander function $w(\theta)$, which facilitates the robot's escape from local minima, providing an alternative direction to pursue if the goal point is currently inaccessible. A new favoured direction θ_w is randomly

generated every 15 seconds. θ_w is limited to a three-quarter-circle range (by setting constant $\alpha = 0.75$) centred on θ_g:

$$\theta_w = adjust\left(\theta_g + 2\pi\alpha\left(rand - \tfrac{1}{2}\right)\right) \tag{5.17}$$

Limiting the range of possible directions causes the robot to travel in the general direction of the goal-point over time (and rarely directly away from it). This range is also large enough to allow the robot to escape from almost any local minimum, given sufficient time. The value of $w(\theta)$ is calculated from θ_w using Equation 5.15 (substituting θ_w for θ_g).

5.2.1.7 Path Following Function

Once combined using Equation 5.10, the avoidance, goal seeking, angular inertia and wander functions form a reactive controller that can navigate through relatively complex environments without the aid of deliberative path planning. However, planning can improve the controller's reliability and speed of goal convergence. Path following function $p(\theta)$ is the interface between deliberative path planning and reactive control. It favours directions that move the robot towards the planned path (or along the path, if the robot is already following it).

First, the distance from the robot's position to each node on the path is calculated, and the closest node is selected (Figure 5.18). Next, the direction θ_p of a node that is distance d_L ahead of the selected node is obtained. The value of d_L affects how closely the robot adheres to the planned path. Finally, $p(\theta)$ is calculated from θ_p in the same manner as $g(\theta)$ is from θ_g and $w(\theta)$ is from θ_w.

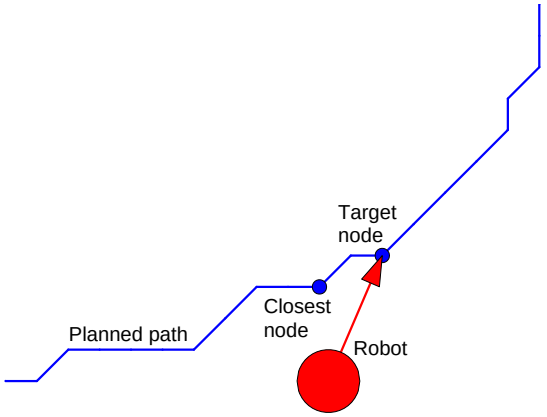

Figure 5.18: Obtaining a target direction θ_p from the planned path.

5.2.1.8 Goal Proximity Modulations

As the robot nears its goal position, the weight of the goal seeking objective is increased, while avoidance, angular inertia and wander weights are reduced. Without this condition, the robot sometimes fails to approach a target position that resides near a wall or other obstacle, because obstacle avoidance is normally the dominant response. At other times the robot might circle a goal point due to the influence of the angular inertia and wander functions, which rarely turn it directly towards the goal.

If the distance d_g to the goal point is within threshold distance $d_{g(\max)} = 1.5$ m, the goal seeking weight W_g is increased linearly as a function of the normalised distance:

$$W_g \leftarrow W_g + \left(1 - \frac{d_g}{d_{g(\max)}}\right)\left(1 - W_g\right) \quad \text{if } d_g < d_{g(\max)} \tag{5.18}$$

The other weights W_n are reduced linearly:

$$W_n \leftarrow W_n \frac{d_g}{d_{g(\max)}} \quad \text{if } d_g < d_{g(\max)} \tag{5.19}$$

5.2.1.9 Capabilities and Limitations

The resulting directional controller is extremely flexible, and it can be tuned to approach the navigation problem using a number of different strategies, depending on its task, its environment and the resources at its disposal. Other behaviours such as wall-following or autonomous docking can be incorporated into the system simply by adding new objectives to the objective function. It is also possible to add multiple goal positions or paths.

In this respect, the control method resembles a traditional behaviour-based system such as Brook's subsumption architecture (e.g. Brooks, 1986). The subsumption architecture decomposes the control problem into discrete layers operating in parallel. A winner-takes-all approach to behaviour selection is employed, with high level layers overriding low level ones. In contrast, our system has the advantage that multiple behaviours simultaneously contribute to motor outputs without conflict.

One moderately similar approach is Arkin's motor schema behaviour coordination mechanism (Arkin, 1989), which allows multiple behaviours to simultaneously contribute to a robot's motor outputs by performing a weighted summation of their output vectors. However, motor schemas each produce a single output vector, rather

than a vector field. This is considerably less information with which to make an informed decision. The approach suffers from the same problems as other artificial potential field navigation methods, such as a tendency towards oscillatory and unstable motion in narrow corridors (Borenstein and Koren, 1991).

Our directional controller produces a locally-optimal direction for the robot to travel. For this direction to result in actual motion, it must be converted into appropriate motor outputs. Many directional obstacle avoidance approaches do not thoroughly consider a robot's motion dynamics when computing its wheel velocities, limiting their high-speed performance in dynamic environments (Fox et al., 1997). Linear velocity is typically a function of the magnitude of a directional controller's output vector, causing a robot to slow down when it approaches an obstacle head-on (Borenstein and Koren, 1991). However, this does not take into consideration a robot's current velocity, motion constraints, or curved trajectories.

In contrast, our planning and control architecture includes an additional stage that gives velocity control similar treatment to directional control. The difference is the conceptual space in which the problem is addressed.

5.2.2 Velocity Control

To obtain a pair of locally-optimal velocities for the robot, the reactive control problem is reformulated as an optimisation in velocity space. The robot's kinematic and dynamic constraints are taken into account, limiting the controller's choice of velocities at any given time to those that the hardware can actually achieve. This approach allows robots to travel at higher speeds than purely directional methods, and it is particularly useful for low-cost robots with limited motor torques (Fox et al., 1997) or for robots with tricycle or quad-wheel drive systems that are unable to perform stationary turns (Lee-Johnson et al., 2007). Two of the most significant examples of this approach are the curvature-velocity (Simmons, 1996) and dynamic window (Fox et al., 1997) obstacle avoidance frameworks.

The curvature-velocity method (Simmons, 1996) represents the robot's trajectory associated with each pair of velocities as a circular arc, or curvature. Obstacles are also represented as circles. The distance to collision along a given curvature can be calculated from the points of intersection between circles. A piecewise curvature-

distance function is pre-calculated for each obstacle. The resulting set of curvature intervals and obstacle distances are utilised to solve a constrained optimisation problem in velocity space. An objective function f associated with velocity couplet (v,ω) is calculated from a linear weighted sum of three objectives:

$$f(v,\omega) = \alpha_1 speed(v) + \alpha_2 dist(v,\omega) + \alpha_3 head(\omega) \tag{5.20}$$

The weights are α_1, α_2 and α_3. The function *speed* favours higher linear velocities, while *dist* favours larger distances to collision, and *head* favours headings closer to the direction of the goal point.

One problem with this method is that it assumes the robot will continue along the selected curvature for some time. However, the robot may only be on the curvature for a single control cycle before choosing a different one. A robot travelling at high speeds may be limited to small directional changes at any given time. By only considering curvatures available in the next control cycle, the robot is likely to miss openings or continue towards obstacles even if there are clear paths around them. This shortcoming is addressed by two approaches that combine directional and curvature-velocity methods. The lane-curvature method (Ko and Simmons, 1998) constructs lanes of free space parallel to the goal heading. Curvatures that turn the robot towards free lanes are favoured by the controller. Similarly, the beam-curvature method (Benayas et al., 2002) constructs radial beams of free space that are favoured during curvature selection.

The dynamic window approach (Fox et al., 1997) solves the optimisation problem by searching a discrete rectangular set of admissible linear and angular velocities. These velocities form a window centred on the robot's current velocities, with borders defined by the robot's velocity and acceleration constraints. The search space is further reduced by eliminating velocities that are likely to result in a collision. This approach employs the same basic objective function as curvature-velocity method. In its original implementation, each obstacle is represented as a line segment whose length is a function of its distance from the robot and the sensor coverage angle. The estimated distance to collision is computed from a line-curvature intersection.

One extension to the dynamic window method incorporates map data in addition to sensor measurements to facilitate the avoidance of obstacles that cannot be easily detected by laser, infrared and ultrasonic sensors (Fox et al., 1998). Others focus on

applying the dynamic window approach to the problem of global motion planning (Brock and Khatib, 1999), (Ögren and Leonard, 2005). However, these methods are not purely reactive in the strictest sense, so they are beyond the scope of this section.

5.2.2.1 Objective Function

Our velocity controller utilises an objective function with the same basic form as that of the directional controller, shown in Equation 5.10. The target linear velocity v_T and target angular velocity ω_T are obtained by applying the objective function $f(v,\omega)$ to a discrete set of admissible velocities (v,ω):

$$\begin{bmatrix} v_T \\ \omega_T \end{bmatrix} = \arg\max_{v\in(v_1,v_2),\omega\in(\omega_1,\omega_2)} f(v,\omega) \tag{5.21}$$

The constraints v_1, v_2, ω_1 and ω_2 are the minimum and maximum linear and angular velocities achievable given the robot's current velocities, acceleration constraints and global velocity limits. They form the boundaries of a rectangular dynamic window.

Unlike the original dynamic window method, hard constraints are not uniformly placed on velocities if they will inevitably lead to a collision. Hard constraints can lead to problems if a situation arises where all available velocities are considered dangerous (e.g. a person suddenly moves in front of the robot while it is travelling at high speed). In such situations, it is advantageous to be able to choose the 'best bad option' (e.g. lower the robot's speed to v_1 and turn sharply away from an obstacle to lessen or avoid the impact). Unlike the dynamic window method's weighted sum objective function, our weighted product objective function effectively prevents subordinate objectives (e.g. speedup) from causing the robot to select an action when a dominant objective (e.g. obstacle avoidance) indicates that the action is highly unfavourable (refer to Figure 5.11 and Figure 5.12 for details). This is functionally similar to a hard constraint during normal operation, but it allows for graceful failure in extreme situations.

5.2.2.2 Avoidance Functions

The objective function includes both sensor-based and map-based obstacle avoidance functions, but aside from the method employed to obtain obstacle positions (described in Section 5.2.1.3), they are functionally identical. Like the curvature-velocity and

dynamic window methods, the robot's anticipated trajectory is represented as a circular arc. However, unlike those methods, each candidate velocity couplet is not rated by the collision-free arc distance of a whole curvature.

Instead, a curvature segment is constructed from the robot's anticipated trajectory over a finite time interval (t_1,t_2), and the distance between this curvature segment and its closest obstacle is calculated (Figure 5.19 and Figure 5.20). The obstacle's radius is not subtracted from this distance, because it is taken into consideration by a subsequent calculation (Equation 5.22). This approach is employed because the robot is generally on a given curvature for a small interval of time before moving onto a different one, so a large portion of the curvature is of little relevance to the selection of velocities in the current control cycle. Methods that consider arc distances to collision over an entire curvature may select certain inferior curvatures that pass very close to an obstacle in the short term, and reject superior ones that are safer in the short term (but more dangerous if the robot stays on them). By measuring obstacle distances only for the portion of the curvature that is considered relevant, the robot is more likely to select safe velocities in certain situations.

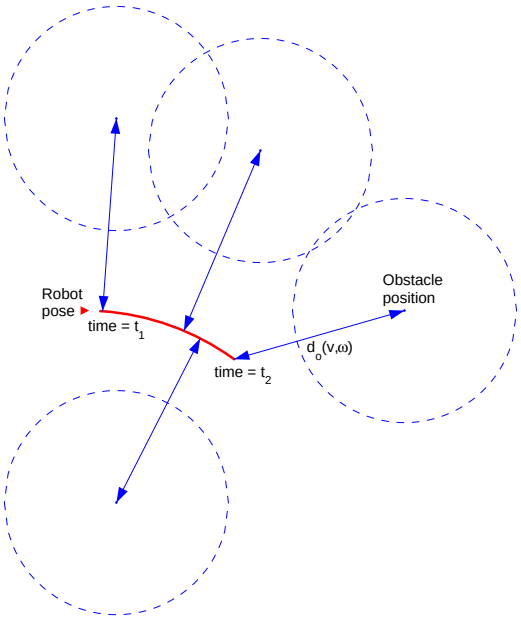

Figure 5.19: Obtaining obstacle distance from the curvature segment (red line) and obstacles. The current position of the centre of the robot (and its current heading) is marked by the red triangle. The dashed blue circles represent the obstacle sizes if they are enlarged by the robot's radius.

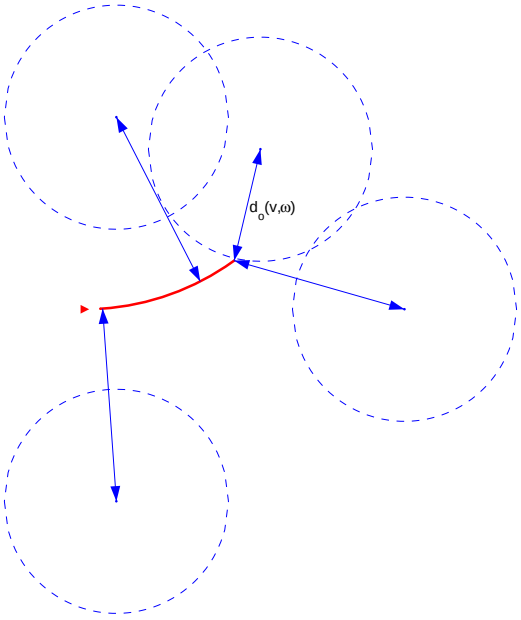

Figure 5.20: A smaller obstacle distance resulting from a less optimal curvature.

The size of a curvature segment is determined by the values of t_1 and t_2. The time t_1 is typically set to one control period (0.1 s). There is little reason for it to be any lower, because the robot is considering its future positions, not its current one. If it is higher, the robot will ignore a larger portion of the curvature when making its decision. The more important parameter is t_2, which directly affects the quality of the trajectory followed by the robot. If it is too high, the robot will be overly conservative in its decisions, rejecting some valid curvatures. Too low, and it may fail to take into account the time required to slow down and avoid a collision.

An angular avoidance value $a_\omega(v,\omega)$ is obtained from the distance $d_o(v,\omega)$ to the obstacle closest to the curvature:

$$a_\omega(v,\omega) = \begin{cases} \kappa \dfrac{d_o(v,\omega)}{r_v} & \text{if } d_o(v,\omega) < r_v \\ \kappa + (1-\kappa)\sqrt{\dfrac{d_o(v,\omega) - r_v}{d_{v(\max)} - r_v}} & \text{else if } d_o(v,\omega) < d_{v(\max)} \\ 1 & \text{otherwise} \end{cases} \tag{5.22}$$

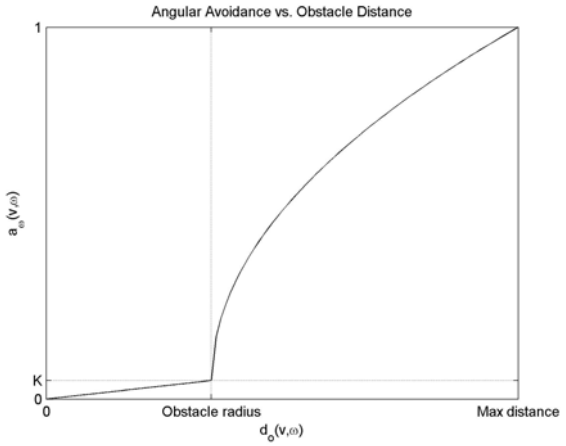

Figure 5.21: Obtaining the angular avoidance value from the measured obstacle distance.

Figure 5.21 shows the structure of this relationship. An implicit obstacle radius is represented by r_v (a separate variable from its directional equivalent r_o). The distance $d_{v(\max)}$ is the threshold beyond which changes in $d_o(v,\omega)$ cease to have an effect (also separate from its directional equivalent $d_{o(\max)}$). Curvatures that pass through obstacles (i.e. those with an obstacle distance less than the radius r_v) are represented in the linear section of the piecewise function. The constant κ controls the maximum objective value of these curvatures. It is typically a non-zero value so that the best velocities can be selected even if all available curvatures pass through an obstacle. A value of $\kappa = 0.05$ is sufficient to allow graceful failure in extreme situations, without adversely affecting performance during normal operation. The nonlinear section of the piecewise function results in a sharp discontinuity that separates curvatures that are highly likely to result in a collision from those that are not. A strong aversive response accompanies curvatures that pass very close to obstacles, with aversion greatly decreasing as the obstacle distances approach a 'safe zone'.

Next, a linear avoidance value $a_v(v)$ is obtained in order to reduce the appeal of high-speed trajectories that pass close to obstacles. A favoured maximum linear velocity $v_{\max}$ is calculated from $a_\omega(v,\omega)$ and the linear velocity limit v_L:

$$v_{\max} = \sqrt{a_\omega(v,\omega)}v_L \tag{5.23}$$

The square-root relationship is employed to ensure that severe reductions are only applied to curvatures that are very close to obstacles. If a candidate velocity exceeds the favoured maximum $v_{\max}$, the reduction of $a_v(v)$ is proportional to the separation between the two values:

$$a_v(v) = \begin{cases} \dfrac{v_L - v}{v_L - v_{\max}} & \text{if } v > v_{\max} \\ 1 & \text{otherwise} \end{cases} \tag{5.24}$$

The combined avoidance function $a(v,\omega)$ is the product of $a_\omega(v,\omega)$ and $a_v(v)$:

$$a(v,\omega) = a_\omega(v,\omega)a_v(v) \tag{5.25}$$

Figure 5.22 shows avoidance values obtained in a large velocity space for the obstacle configuration shown in Figure 5.19 and Figure 5.20. At any given moment, the robot's current velocities and acceleration constraints limit the available velocities to a small subset of those shown. All possible trajectories pass relatively close to obstacles, so the robot cannot approach the maximum speed (1 m/s) without risking a collision. Right turns are generally favoured over left turns, due to the three obstacles on the robot's left.

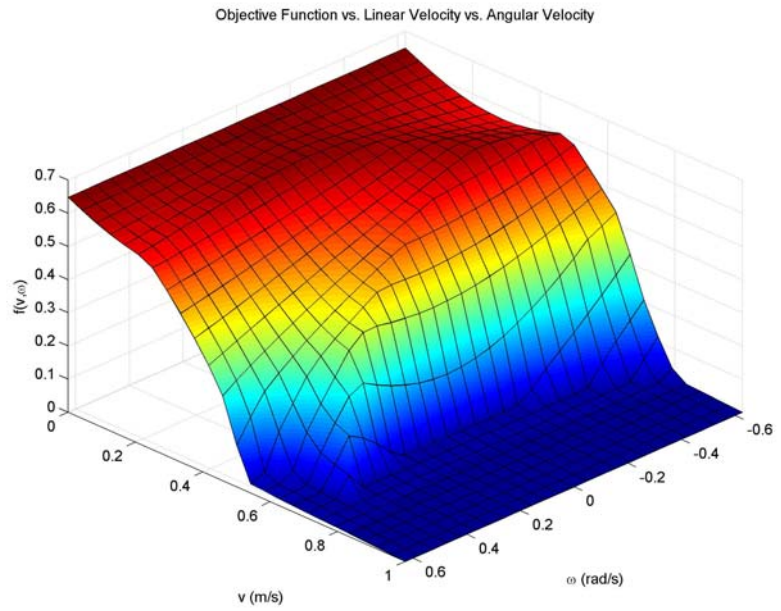

Figure 5.22: Avoidance function applied to a large velocity space. Variables: t_1 = 0.1 s, t_2 = 1.0 s, κ = 0.05, $d_{v\,(\max)}$ = 1 m.

5.2.2.3 Goal Seeking Function

The avoidance function includes no incentive to follow a goal direction or favour higher speeds over lower ones, so it is unable to produce viable motion on its own. The next main component of the objective function is the goal seeking function

$g(v,\omega)$, which encourages the robot to move in the directional controller's favoured direction.

First, an angular goal seeking value $g_\omega(\omega)$ is obtained that favours angular velocities that turn the robot towards the target heading θ_T received from the directional controller:

$$g_\omega(\omega) = 1 - \frac{|adjust(\theta_T - \theta_P(\omega))|}{\pi} \tag{5.26}$$

The angle $\theta_P(\omega)$ is the robot's predicted heading for a given angular velocity ω after time t_3, given the robot's current heading θ_C:

$$\theta_P(\omega) = \theta_C + \omega t_3 \tag{5.27}$$

Thus, the time t_3 directly influences the relationship between $g_\omega(\omega)$ and ω. Low values of t_3 result in little variation in $g_\omega(\omega)$ with different ω values when the heading error is high, causing $g_\omega(\omega)$ to have a comparatively low influence on velocity selection. However, when the heading error is high, low values of t_3 cause the robot to fine-tune its heading precisely to match the target heading. In both situations, the reverse is true for high values of t_3.

Second, a linear goal seeking value $g_v(v)$ is obtained that favours smaller linear velocities (and therefore near-stationary turns). A favoured maximum linear velocity $v_{\max}$ is derived from the linear velocity limit v_L and angular error:

$$v_{\max} = \begin{cases} \left(1 - \dfrac{|adjust(\theta_T - \theta_C)|}{\beta\pi}\right) v_L & \text{if } |adjust(\theta_T - \theta_C)| < \beta\pi \\ 0 & \text{otherwise} \end{cases} \tag{5.28}$$

Velocity reductions are proportional to the heading error until the angular error exceeds a threshold controlled by the turning factor β, at which point $v_{\max}$ reaches zero. The value of β determines the strength of the incentive to slow down when the robot is not facing the intended direction. Low values result in erratic motion as the robot constantly stops to make course corrections. Higher values generally yield smoother motion, but if β is too high, the resulting large circular turns can cause significant deviations from the expected path and potential failure to converge on the goal point. Smaller linear velocities resulting from low β values are also advantageous for preventing the robot from overshooting doorways.

Equation 5.24 is employed to obtain $g_v(v)$ from v_{max} (replacing $a_v(v)$ with $g_v(v)$). Finally, the angular and linear goal seeking values are combined to form the overall goal seeking function $g(v,\omega)$:

$$g(v,\omega) = g_\omega(\omega) g_v(v) \tag{5.29}$$

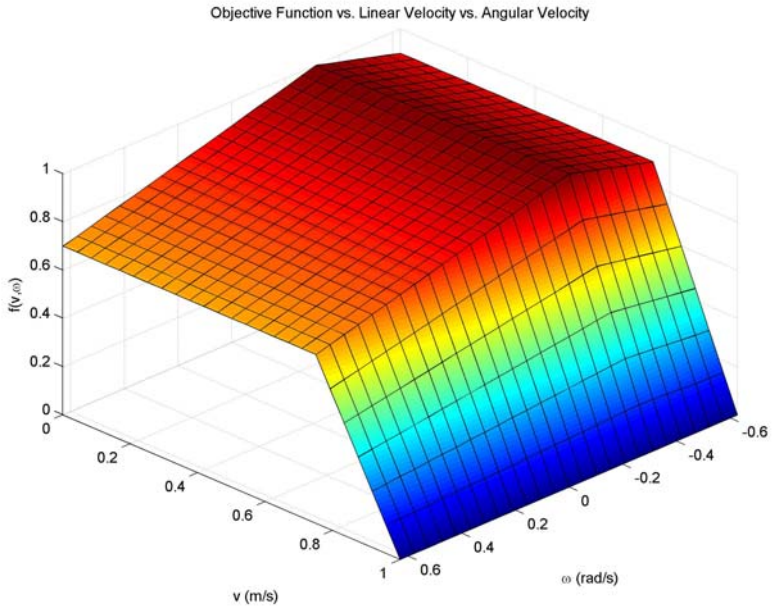

Figure 5.23: Goal seeking function. Variables: θ_T = -0.1π, t_3 = 1 s, β = 0.4.

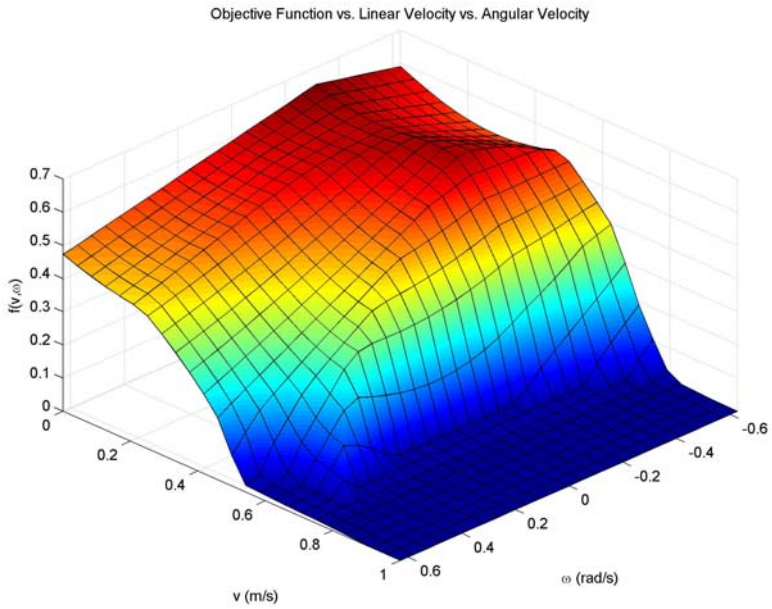

Figure 5.24: Objective function combining avoidance and goal seeking functions. Variables: Sensor-based avoidance weight W_{v1} = 1, goal seeking weight W_{v3} = 0.9.

By applying the goal seeking function to a target heading of -0.1π radians, we get the form shown in Figure 5.23. In this example, the heading error is relatively small, so only the highest linear velocities are assigned reduced priority due to $g_v(v)$. An objective function resulting from a combination of avoidance and goal seeking functions is shown in Figure 5.24. The function retains much of the avoidance function's shape, but angular velocities are prioritised by their proximity to the optimal value of ω = -0.1π rad/s.

5.2.2.4 Speedup Function

An objective function that includes both avoidance and goal seeking functions causes the robot to favour obstacle-free trajectories that move it in the goal direction. However, the robot is unlikely to start moving, because the safest trajectories tend to be those that result in no forward motion. A general incentive to move is provided by the speedup function $s(v)$, which favours high linear velocities. Candidate linear velocities are biased according to the following equation:

$$s(v) = \frac{v}{v_L} \tag{5.30}$$

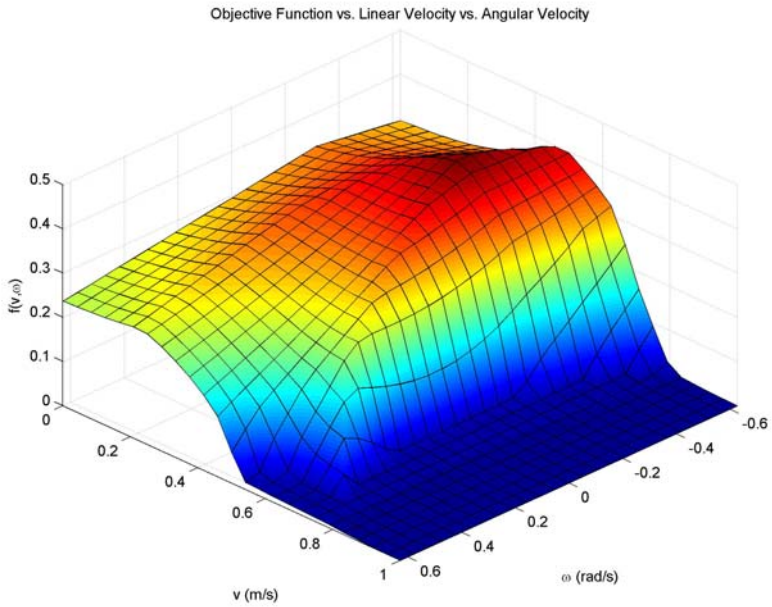

Figure 5.25: Objective function combining avoidance, goal seeking and speedup functions. Variables: Sensor-based avoidance weight W_{v1} = 1, goal seeking weight W_{v3} = 0.9, speedup weight W_{v4} = 0.5. The velocities that yield the highest objective function value are selected for the motor outputs.

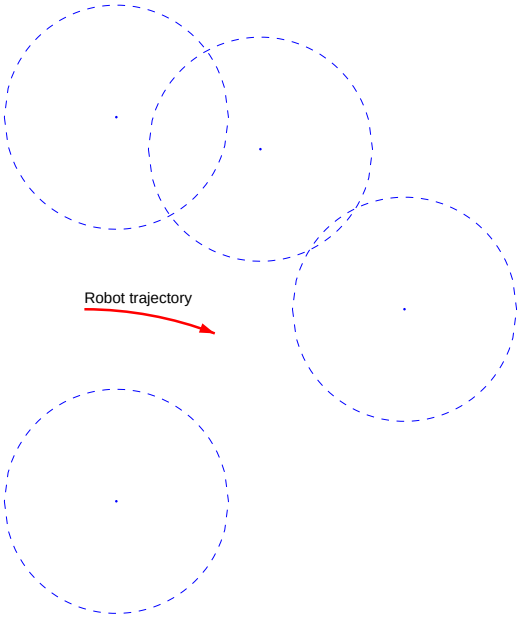

Figure 5.26: Robot's trajectory if it maintains velocities of v = 0.42 m/s, ω = -0.37 rad/s for 1 second.

Figure 5.25 shows an objective function containing avoidance, goal seeking and speedup functions applied to the previous scenario. By reducing the appeal of the lower velocities, the robot is finally compelled to move forward in addition to avoiding obstacles and orienting itself towards the goal direction. The optimal velocities in this example result in the trajectory shown in Figure 5.26.

5.2.2.5 Distance-to-Goal Function

Avoidance, goal seeking and speedup functions are sufficient to produce viable motion throughout most of a journey. A problem arises when the robot approaches the goal point, however. Lacking an incentive to slow down when it reaches its target, the robot tends to overshoot or circle the goal point. This problem is solved by the distance-to-goal function $d(v)$, which facilitates convergence to the goal point by reducing the robot's speed. The corresponding maximum linear velocity is a function of the distance d_g from the target position:

$$v_{\max} = \begin{cases} \dfrac{d_g}{d_{g(\max)}} v_L & \text{if } d_g < d_{g(\max)} \\ v_L & \text{otherwise} \end{cases} \tag{5.31}$$

The objective value $d(v)$ is obtained from v_{max} by Equation 5.24 (replacing $a_v(v)$ with $d(v)$).

5.2.2.6 Goal Proximity Modulations

Like the directional controller, the velocity modulates certain control weights as the robot nears its goal position. The goal seeking weight is increased by Equation 5.18 and the speedup weight is reduced by Equation 5.19. Failure to modulate the goal seeking weight in this manner renders low values unviable. The robot tends to fail to converge on the goal point, because the avoidance function's incentive to turn the robot away from obstacles outweighs the goal seeking function's incentive to approach the goal. Similarly, if the speedup weight is not modulated, high values cause the robot to fail to converge on the goal point. This is because the speedup function's incentive to choose higher speeds outweighs the distance-to-goal function's incentive to slow down.

5.2.2.7 Capabilities and Limitations

At the core of the velocity controller is an objective function with the same form as that utilised by the directional controller. Essentially, it provides a unified approach to solving problems in two different conceptual spaces. Of the two systems, the directional controller has a more global perspective (although it is still highly reactive compared to a deliberative planner). It can consider larger displacements in time and space, but it ignores robot motion dynamics and thus cannot ensure safety. Conversely, the velocity controller has a much narrower conceptual view, focusing on motion that the robot can achieve safely over a very short time interval. This is not necessarily an inherent limitation of velocity space methods in general. Our velocity controller actually has a narrower perspective than most other velocity space methods because it is not utilised in isolation, but rather in combination with the directional controller.

Neither controller on its own is ideal for producing optimal collision-free motion in dynamic environments. However, once combined, the two systems form a robust and flexible reactive navigation system. The lane-curvature method (Ko and Simmons, 1998) and beam-curvature method (Benayas et al., 2002) also combine directional and

velocity space techniques, but these approaches are quite different from the one employed in this thesis. They are essentially single-level velocity space methods that incorporate directional control elements. In contrast, our controller incorporates directional and velocity space approaches as distinct components, each with independent obstacle avoidance capabilities.

This approach results in various redundancies, which can be advantageous to a robot controller. Redundancies can facilitate adaptation by allowing a robot to select alternative problem solving strategies, should an individual strategy fail. For example, if the directional controller's obstacle avoidance function prevents it from traversing a narrow doorway, its safety margins or weights could be lowered, causing the robot to rely more heavily on the velocity controller for obstacle avoidance.

5.3 Summary

A hybrid reactive/deliberative planning and control architecture has been developed that supports point-to-point navigation and exploration in arbitrary flat-surfaced environments. It combines deliberative mapping and path planning capabilities with two distinct reactive navigation approaches – directional and velocity space control. The architecture is highly flexible, enabling the robot to adaptively and continuously adjust its navigation strategies to suit its environment, task, momentary situation and the resources at its disposal. This flexibility arises from various planning and control parameters that can be modulated to influence the robot's behaviour either subtly or overtly as required.

In this chapter, few specifics have been presented regarding the exact values of the robot's planning and control parameters. In part, this is because many of them are not constants; they are variables modulated by the robot's affective system. Chapters 6 and 7 will show the effects of various parameters on the robot's performance in a range of simulation experiments. This will allow us to group them into different categories indicating which parameters should be kept constant, which ones can be modulated, and to what effect.

6 Basic Navigation Experiments

The various components of our planning and control architecture have been described in Chapter 5. To demonstrate the effectiveness of this architecture and verify the contributions of the various components, a series of experiments are conducted in a range of procedurally-generated environments. Control layers are tested from the bottom up, starting with the velocity controller, followed by the directional controller, and finally the deliberative path planner. This typically involves varying control weights to incrementally increase or decrease the contributions of individual components.

Attempting to thoroughly test all configurations under all conceivable conditions would be a computationally intractable problem. Instead, individual weights are varied incrementally, while the remainder of weights and other parameters are kept constant in configurations that are anticipated to produce satisfactory performance.

A set of 20 environments is generated procedurally. Different environments are utilised for each of the 20 individual test runs within a given experiment, but those same 20 environments are reused in the other experiments, to ensure consistency between them. Under each different weight configuration, the robot is instructed to navigate between the same two points (beginning with a heading of 0 radians). All internal data, such as maps or variable states are reset at the beginning of each experimental run – nothing is retained. Performance characteristics such as collision rate and time taken to reach the goal are recorded for each experimental run. If the robot fails to reach the goal within 10 minutes, it times out. In that situation a failure is recorded, the goal completion time is set to 10 minutes, and the experimental run is not repeated.

6.1 Velocity Control

To test the velocity controller independently from the higher control layers, the direction of the goal point is passed directly to its goal seeking function (whereas normally a translated direction would be received from the directional controller). The velocity controller's goal seeking weight W_{v3} and speedup weight W_{v4} are varied,

while the remainder of the robot's weights and parameters are kept constant at values shown in Table 6.1.

TABLE 6.1: PARAMETERS – VELOCITY CONTROLLER EXPERIMENTS

Symbol	Name	Value	Justification
W_{v1}	Sensor-based avoidance weight	0.99	Highest allowable value, to minimise collisions.
W_{v2}	Map-based avoidance weight	0	Weight is irrelevant because there is no deliberative map.
W_{v5}	Distance-to-goal weight	0.9	High enough to outweigh the speedup weight after goal proximity modulations.
(n_v,n_ω)	Size of dynamic window (rows, columns)	(9, 9)	High enough to produce smooth motion; low enough to be computationally efficient. Odd numbers to ensure that the robot's current velocities are included (refer to Section 7.3.1).
(v_L,ω_L)	Velocity limits (linear, angular)	(1 m/s, 0.2π rad/s)	v_L: An estimate of the highest safe value in a static indoor environment. ω_L: Limited in order to minimise disturbances due to sensor filtering.
(a_{max},α_{max})	Acceleration limits (linear, angular)	(0.5 m/s^2, π rad/s^2)	High enough to allow rapid response to environment; low enough to limit internal stresses and to be achievable by the robot's drive system.
(a_{min},α_{min})	Deceleration limits	(-0.5 m/s^2, -π rad/s^2)	Same as above.
n_o	Obstacle buffer size	5	A compromise between safety and efficiency.
(t_1,t_2)	Avoidance curvature times	(0.1 s, 1 s)	High enough to prevent most collisions; low enough not to significantly inhibit motion.
t_3	Goal seeking time	1 s	A compromise between optimal low/high heading error response.
r_o	Obstacle radius	0.35 m	Set to the robot's radius.
$d_{v(max)}$	Maximum obstacle distance (velocity control)	1 m	Encourages robot to keep away from obstacles without inhibiting motion.
β	Turning factor	0.4	A compromise between motion smoothness and deviation from the current position while turning.

In the first experiment, the goal seeking weight W_{v3} is set to 0 while the speedup weight W_{v4} is iteratively varied. With $W_{v3} = 0$, the robot moves aimlessly throughout

the environment. It gives no preference to any particular direction, instead choosing trajectories purely on the basis of collision avoidance (due to the sensor-based avoidance function) and maintaining higher linear velocities (due to the speedup function). However, if it randomly approaches within 1.5 m of the goal point, W_{v3} increases due to goal proximity modulations, causing the robot to converge on the goal. This allows the robot to achieve some success at goal convergence even with no incentive to reach it for most of the journey.

Low values of W_{v4} (particularly those lower than 0.4) tend to prevent the robot from traversing doorways, as the incentive to select higher velocities is insufficient to overcome the avoidance function's incentive to slow down near obstacles (Figure 6.1). High values allow the robot to traverse doorways, but result in a higher incidence of collisions, because the robot approaches closer to obstacles before the avoidance function's incentive to slow down can overcome the incentive for speed (Figure 6.2). Medium values (around 0.5) are the most optimal, allowing the traversal of doorways, but coming at minimal cost to safety (Figure 6.3). Figure 6.4 shows the overall increase in collision rate as W_{v4} increases. When $W_{v4} > 0.7$, the robot's motion is highly unsafe, resulting in numerous collisions.

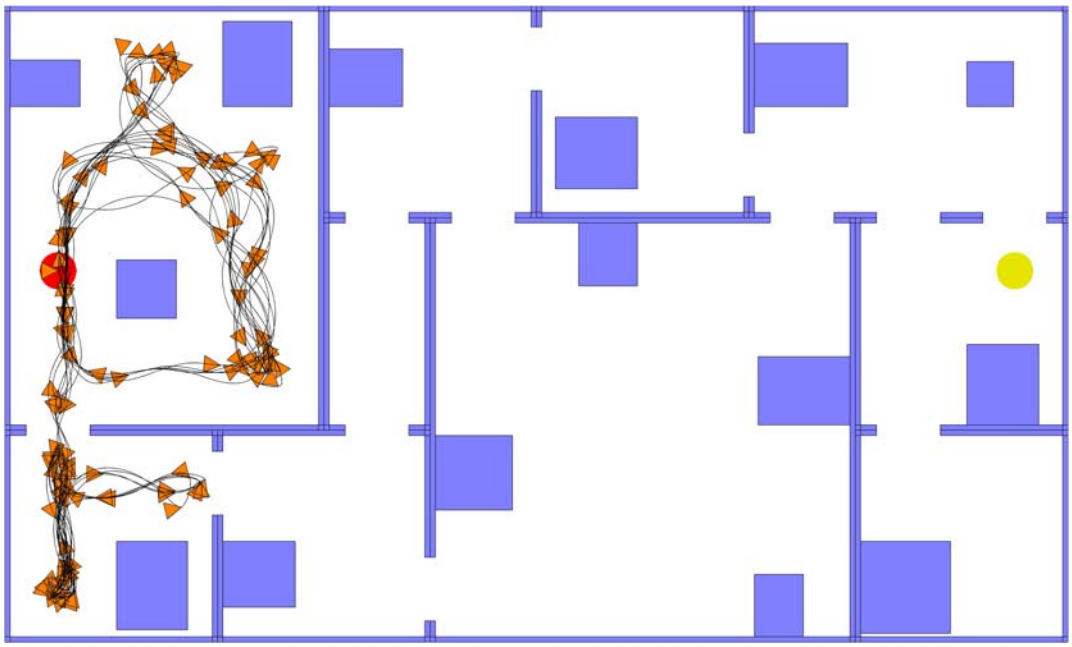

Figure 6.1: Robot's path with W_{v4} = 0.3. The environment shown covers a 20 m × 12 m area.

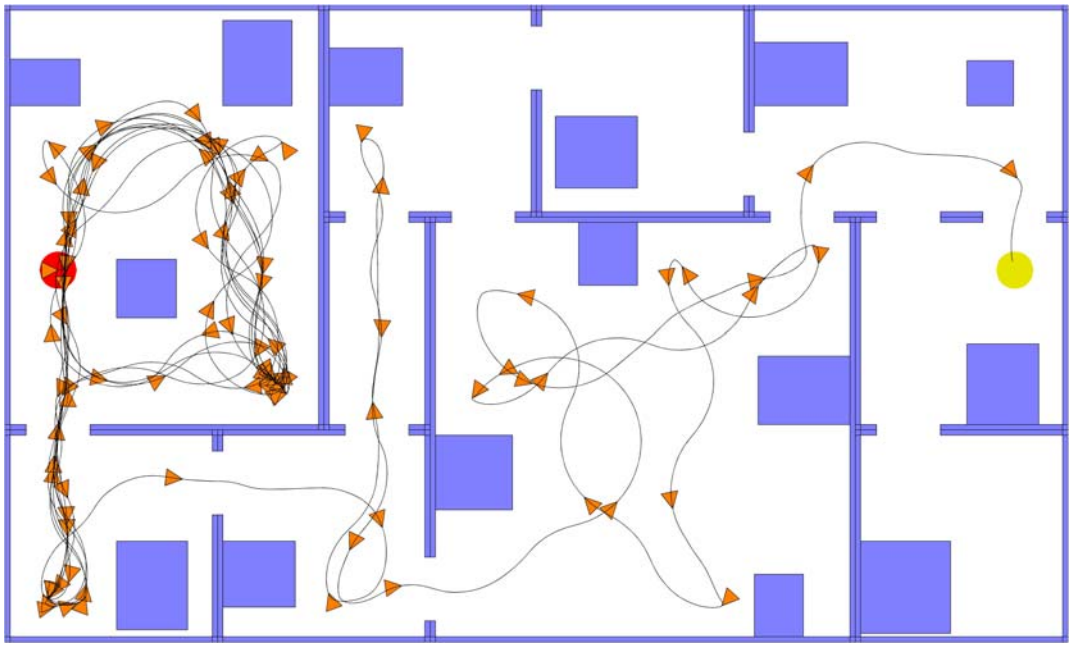

Figure 6.2: Robot's path with W_{v4} = 0.5.

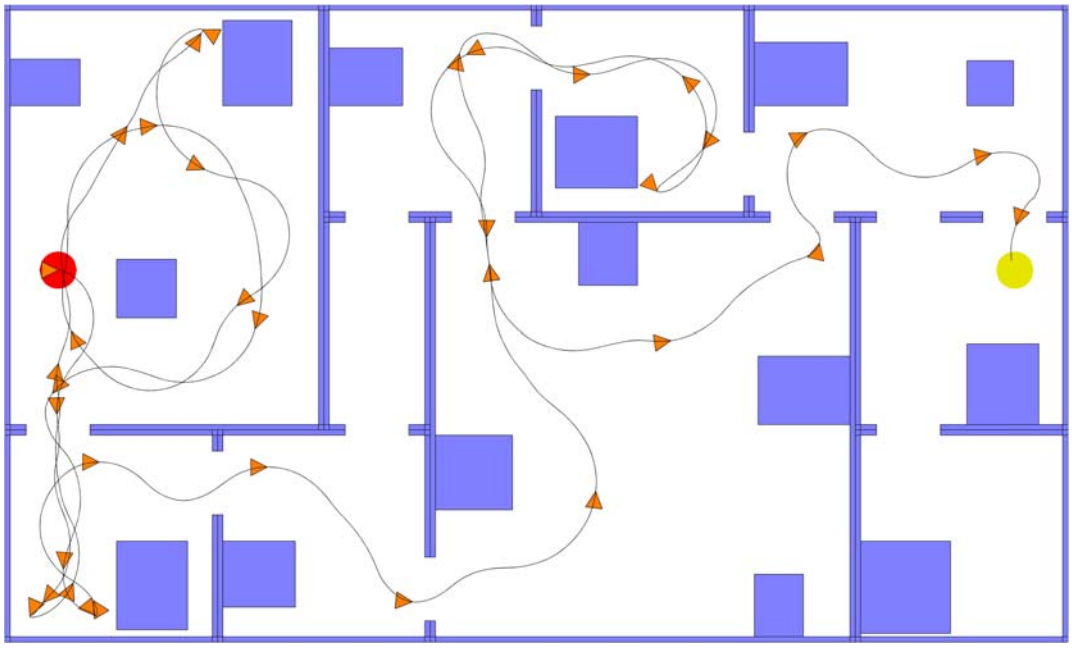

Figure 6.3: Robot's path with W_{v4} = 0.9.

Collisions per minute are recorded rather than total number of collisions, due to the variable completion times for individual experimental runs. For example, a robot that takes 10 minutes to reach its goal and undergoes 4 collisions is considered 'safer' than one that reaches the goal in 5 minutes and sustains 3 collisions.

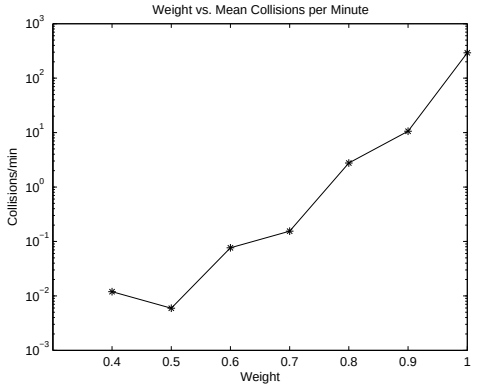

Figure 6.4: Mean collisions per minute for 20 experimental runs per W_{v4} value. Zero collisions occurred when $W_{v4} < 0.4$, so those data points do not appear on the logarithmic scale.

In the second experiment, W_{v3} is varied, while W_{v4} is kept constant at 0.5 (a value that is high enough to allow traversal of doorways, but low enough to minimise the likelihood of collision). Figure 6.5 shows the resulting success rate, or percentage of environments in which the robot reaches the goal before the timeout. The success rate quickly decreases as W_{v3} increases, and no successes are achieved when $W_{v3} > 0.1$. While goal seeking prevents the robot from wandering aimlessly, it also causes the robot to stop moving when certain obstacle configurations block its path to the goal (Figure 6.6).

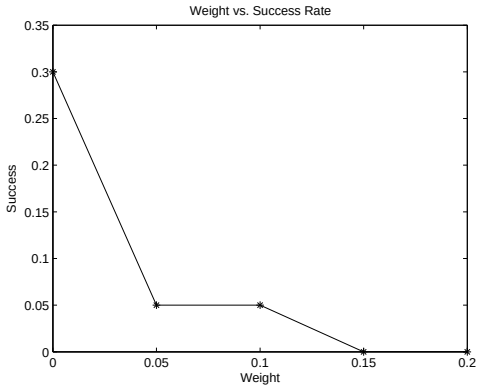

Figure 6.5: Mean success rate for 20 experimental runs per W_{v3} value.

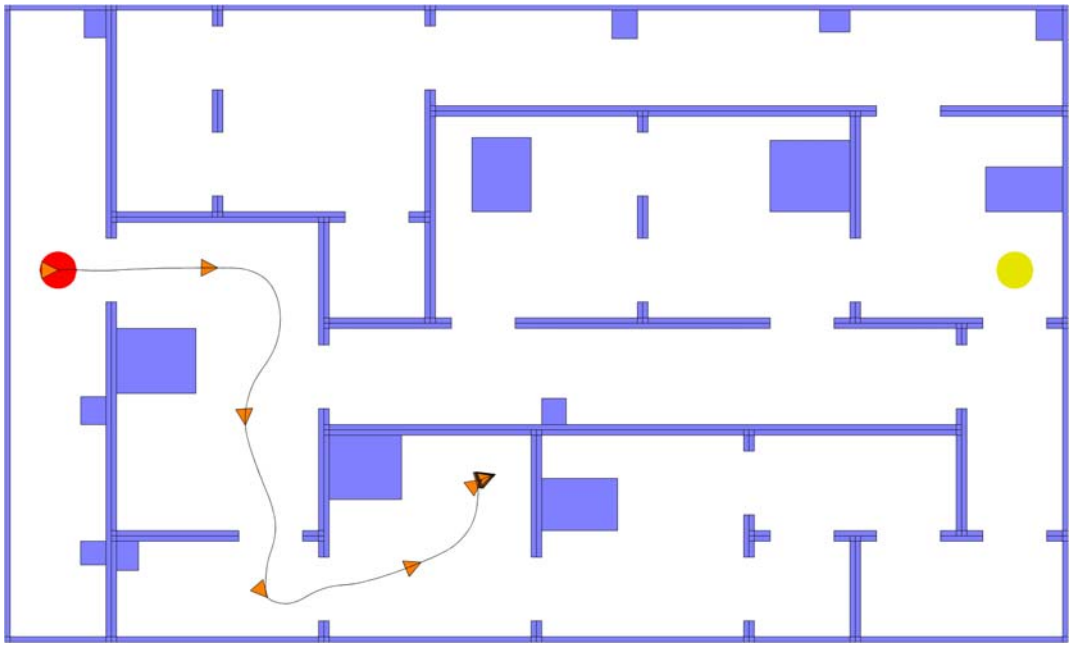

Figure 6.6: Robot's path with W_{v3} = 0.1.

The complex indoor environments in which these experiments are conducted generally pose too great a navigational challenge for the velocity controller on its own. Goal-directed behaviour is the responsibility of the higher control levels, so it is unsurprising that the velocity controller performs poorly in this area.

6.2 Directional Control

The directional controller is dependent on the velocity controller in that it cannot produce motor outputs on its own. However, it can be analysed somewhat independently by disabling the velocity controller's avoidance functions. This produces similar behaviour to a single-stage directional controller, even though it is not explicitly single-stage. Table 6.2 shows the parameter values employed in these experiments, in addition to (or modified from) those shown in Table 6.1.

First, the weight $W_{\theta 5}$ controlling the angular inertia objective is iteratively varied from 0 to 1, while the goal seeking weight $W_{\theta 4}$ = 0.5, and the wander weight $W_{\theta 6}$ = 0 (to eliminate its contribution to behaviour) are kept constant.

TABLE 6.2: PARAMETERS – DIRECTIONAL CONTROLLER EXPERIMENTS

Symbol	Name	Value	Justification
W_{v1}	Sensor-based avoidance weight (velocity)	0	To test the directional controller independently by disabling the velocity controller's obstacle avoidance capabilities.
W_{v3}	Goal seeking weight (velocity)	0.99	To make the velocity controller rigidly obey the directional controller.
W_{v4}	Speedup weight (velocity)	0.5	A compromise between safety and goal-directedness.
$W_{\theta 1}$	Sensor-based avoidance weight (directional)	0.99	Highest allowable value, to minimise collisions.
$W_{\theta 2}$	Map-based avoidance weight (directional)	0	Weight is irrelevant because there is no deliberative map.
$W_{\theta 3}$	Path following weight (directional)	0	To disable the contribution of deliberative path planning.
$W_{\theta 4}$	Goal seeking weight (directional)	0.5	A medium value that provides adequate goal seeking incentive without jeopardising safety.
n_{θ}	Number of vectors in vector field	50	High enough to produce smooth motion; low enough to be computationally efficient.
$d_{o(\max)}$	Maximum obstacle distance (directional)	3 m	Set to the sensor range.
s	Smoothing factor	0.1	Encourages robot to stay away from obstacles without inhibiting motion.

With the angular inertia function disabled (by setting $W_{\theta 5} = 0$), even small concave regions become significant obstructions. There is no incentive for the robot to maintain its heading, so it is highly prone to oscillatory behaviour. In the environment shown in Figure 6.7, there are two regions where the robot becomes trapped. Eventually it manages to escape from the first, but it is unable to escape the second minima before the 10 minute timeout is triggered.

With angular inertia strongly enabled, the robot favours maintaining its current heading over approaching the goal. It tends to continue along its path until it reaches an obstacle, whereupon it generally turns towards the goal. In many environments, this type of behaviour is sufficient to escape from local minima and converge on the goal. The environment in which it previously failed is one such example (Figure 6.8).

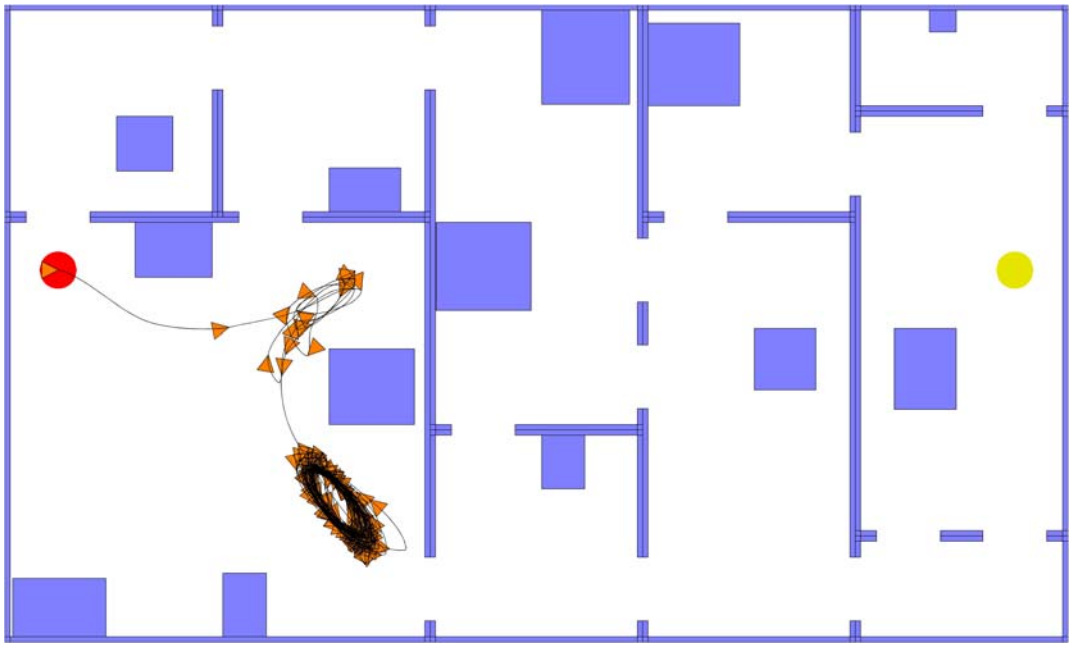

Figure 6.7: An unsuccessful path with $W_{\theta 5}$ = 0.

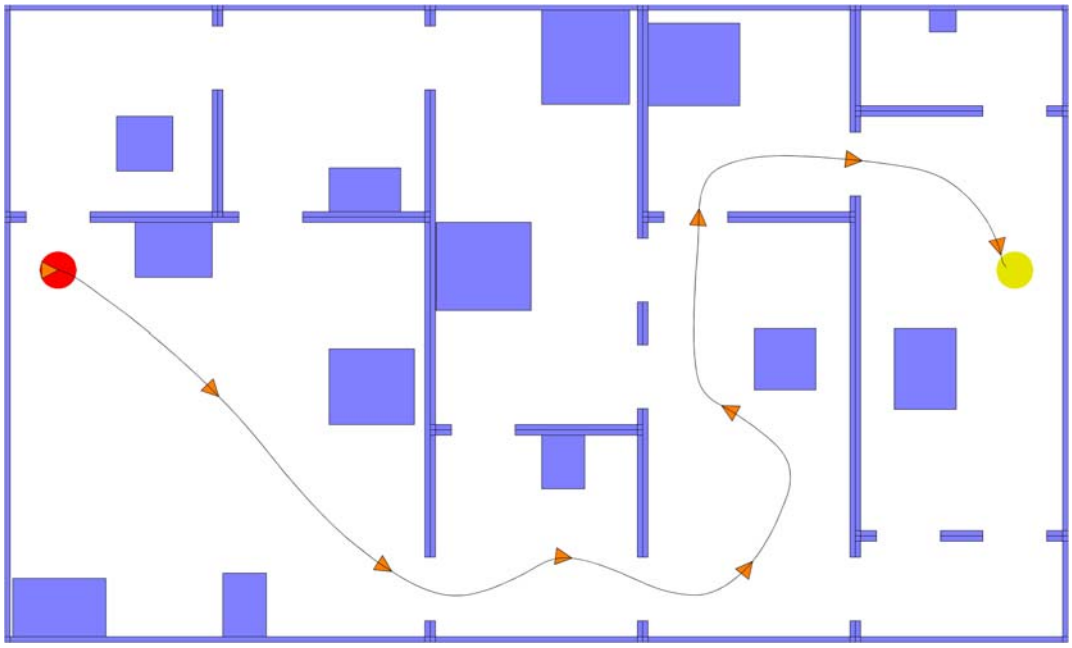

Figure 6.8: A successful path with $W_{\theta 5}$ = 0.8.

However, in certain other environments, angular inertia can cause the robot to become trapped in an infinite cycle, preventing it from ever reaching the goal. One type of environmental feature that can cause this is a room containing a central obstacle whose doorways face away from the goal point, such as that shown in Figure 6.9. This induces a circular pattern of motion that the robot is unable to escape.

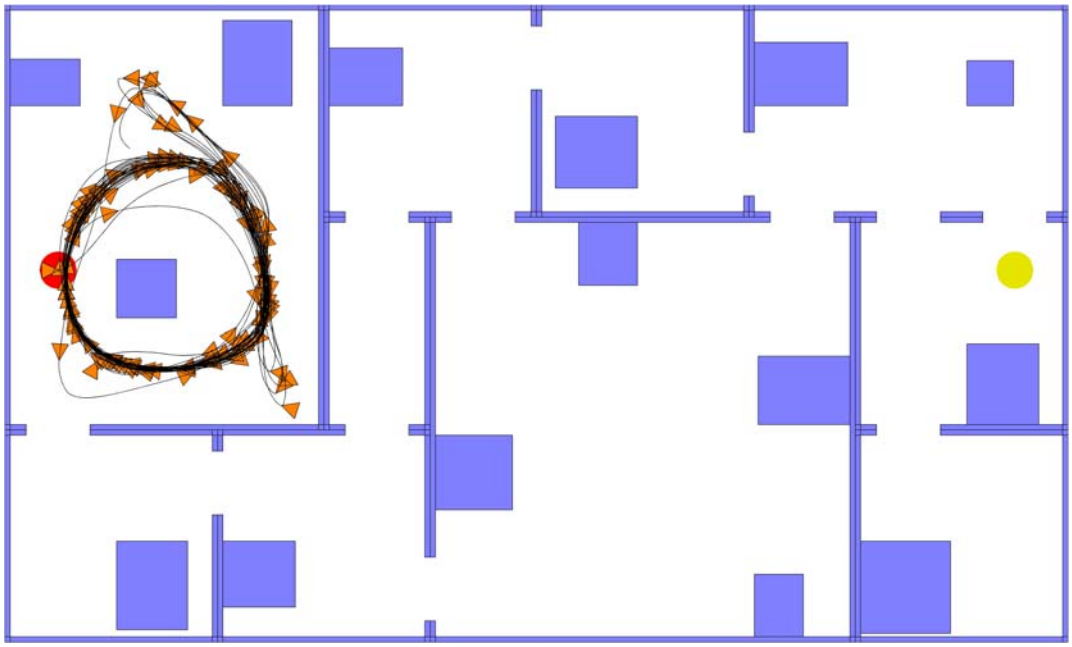

Figure 6.9: An unsuccessful path with $W_{\theta 5}$ = 0.8.

In general, the completion time (Figure 6.10) decreases as $W_{\theta 5}$ increases, reaching a minimum of 349 s when $W_{\theta 5}$ = 1. The success rate (Figure 6.11) shows a corresponding increase, reaching a maximum of 65% when $W_{\theta 5}$ = 1. However, this improvement comes at a significant cost to safety, increasing the collision rate (Figure 6.12). This is because the incentive to maintain the current heading increasingly overcomes the incentive to turn away from obstacles.

Greater success can be achieved by reducing $W_{\theta 5}$ to 0.5, and enabling the wander objective by setting its weight $W_{\theta 6}$ to a non-zero value. As this objective is assigned a higher level of influence, the robot's behaviour becomes increasingly stochastic. This allows it to escape from virtually any local minima, given sufficient time. Although the favoured direction of the wander objective at any given time is random, the range of possible directions is centred on the goal direction. Furthermore, the goal seeking objective is not disabled. Thus, the robot's movement remains biased towards the goal point even when the wander objective is strongly enabled.

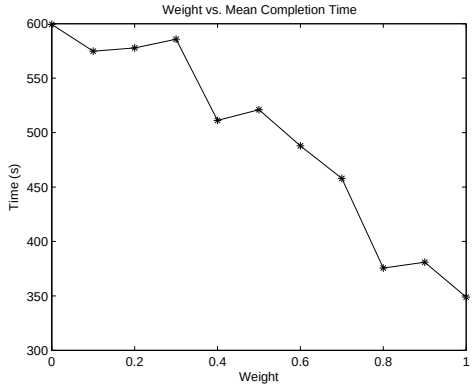

Figure 6.10: Mean completion time for 20 experimental runs per $W_{\theta 5}$ value.

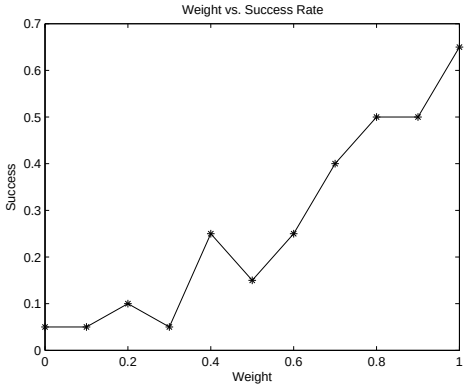

Figure 6.11: Mean success rate vs. $W_{\theta 5}$.

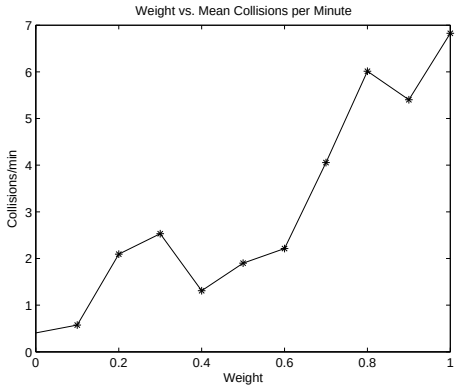

Figure 6.12: Mean collisions per minute vs. $W_{\theta 5}$.

As shown in Figure 6.13, the wandering robot is able to escape from the region that traps the inertia-driven robot. The obvious disadvantage of the wander objective is that the robot can sometimes take a long time to reach its goal, due to its randomness. The environment shown in Figure 6.14 is one example where it fails to reach the goal before the 10 minute timeout.

The completion time shows a steady decrease as $W_{\theta 6}$ increases (Figure 6.15). It reaches a minimum of 272 s when $W_{\theta 6} = 1$, significantly lower than the best mean completion time of the inertia-driven robot. The success rate (Figure 6.16) reaches a peak of 85% at $W_{\theta 6} = 0.7 – 0.8$. This is significantly higher than the robot is able to achieve with angular inertia alone. Again, the improvement to completion time and success rate is accompanied by an increased number of collisions (6.17).

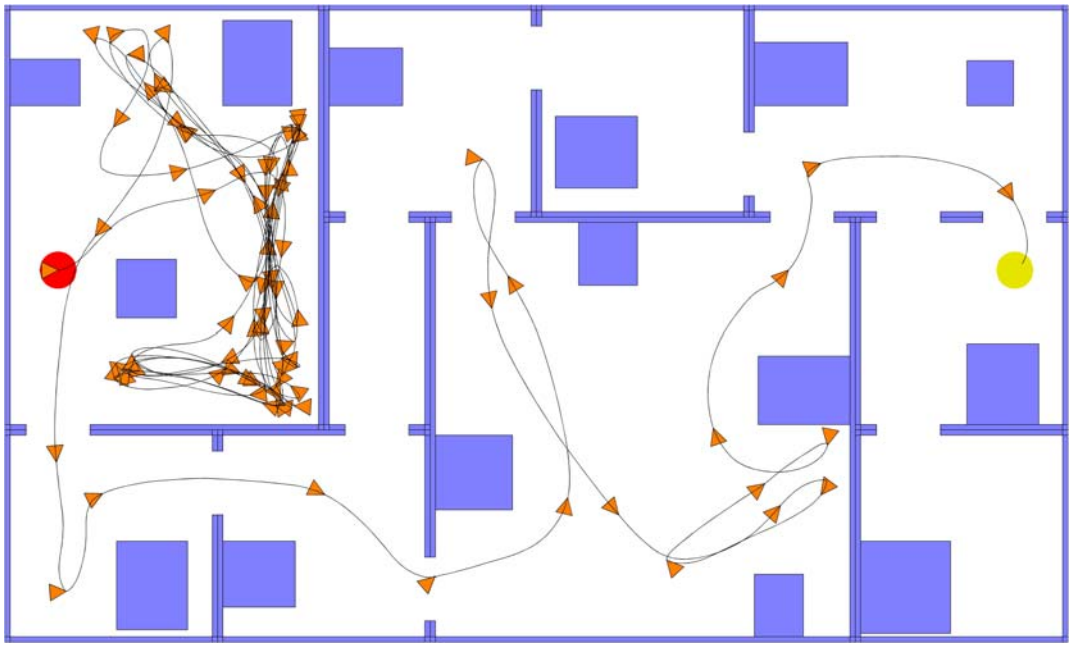

Figure 6.13: A successful path with $W_{\theta 6}$ = 0.8.

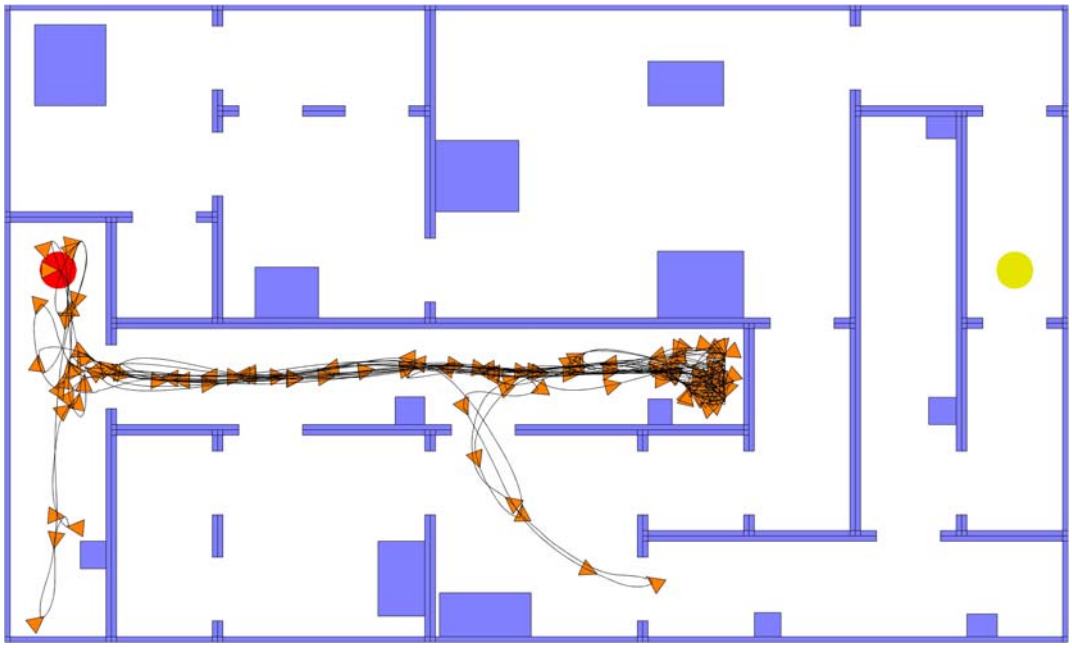

Figure 6.14: An unsuccessful path with $W_{\theta 6}$ = 0.8.

Overall, the directional controller is relatively successful at goal-directed behaviour even with deliberative capabilities disabled. However, its high-speed obstacle avoidance capabilities are inferior to those of the velocity controller.

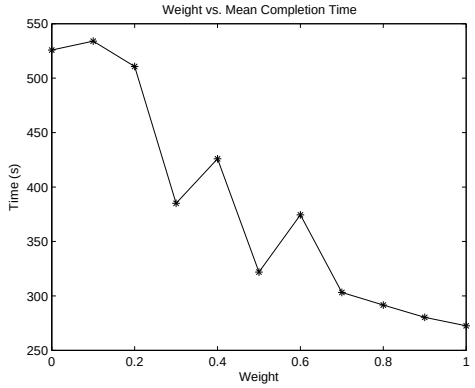

Figure 6.15: Mean completion time for 20 experimental runs per $W_{\theta 6}$ value.

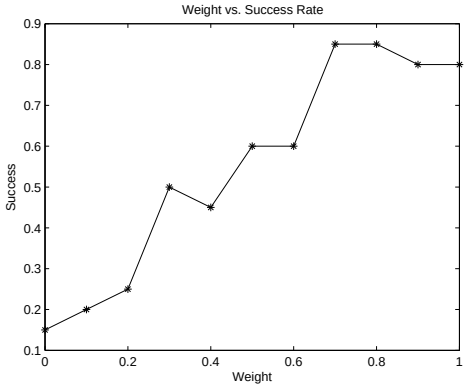

Figure 6.16: Mean success rate vs. $W_{\theta 6}$.

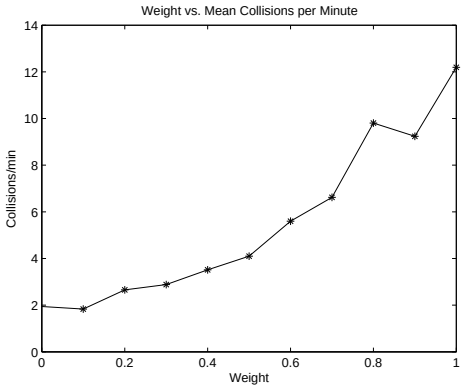

Figure 6.17: Mean collisions per minute vs. $W_{\theta 6}$.

6.3 Two-stage Reactive Control

The advantages of employing a two-stage directional/velocity space method over a purely directional controller are best demonstrated by showing the effects of the velocity controller's avoidance weight W_{v1} and goal seeking weight W_{v3} on performance. Based on the results of the previous experiment, the directional control weights absent from Table 6.2 are set as follows: $W_{\theta 5} = 0.5$, $W_{\theta 6} = 0.75$.

First, W_{v1} is varied, while W_{v3} is set to 0.75 (a satisfactory value, as shown in the subsequent experiment). Under conditions that produce adequate goal seeking behaviour, the directional controller's incentive to stay away from obstacles is too low, resulting in numerous collisions (Figure 6.18). However, when the velocity controller's avoidance function is strongly enabled in conjunction with the directional

controller, satisfactory goal seeking performance can be achieved without the high collision count (Figure 6.19).

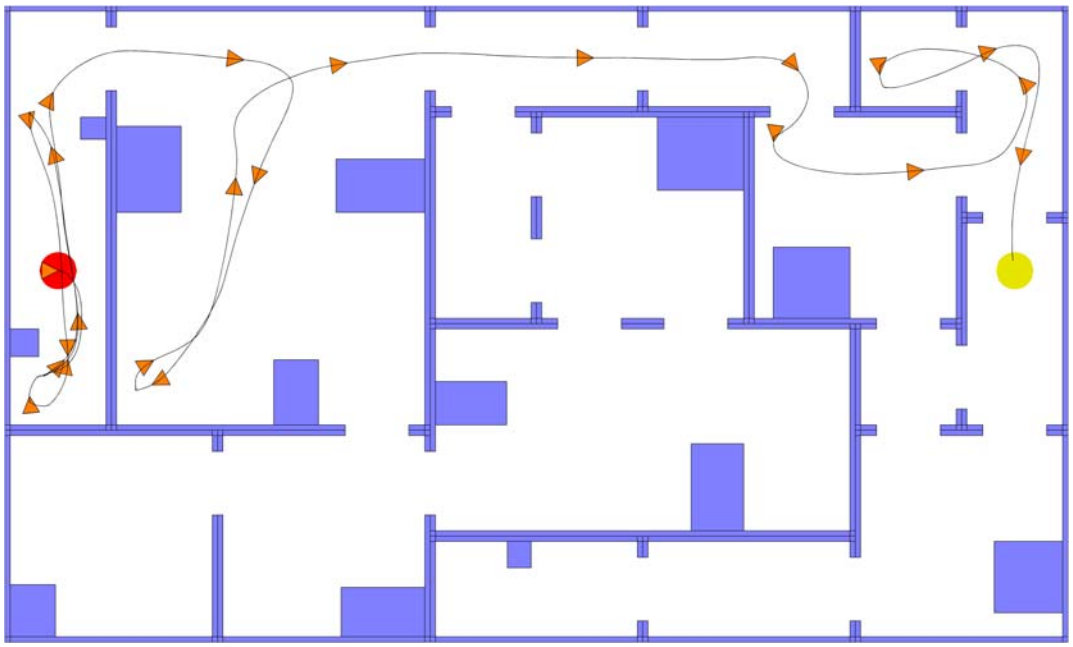

Figure 6.18: The robot's path with W_{v1} = 0.

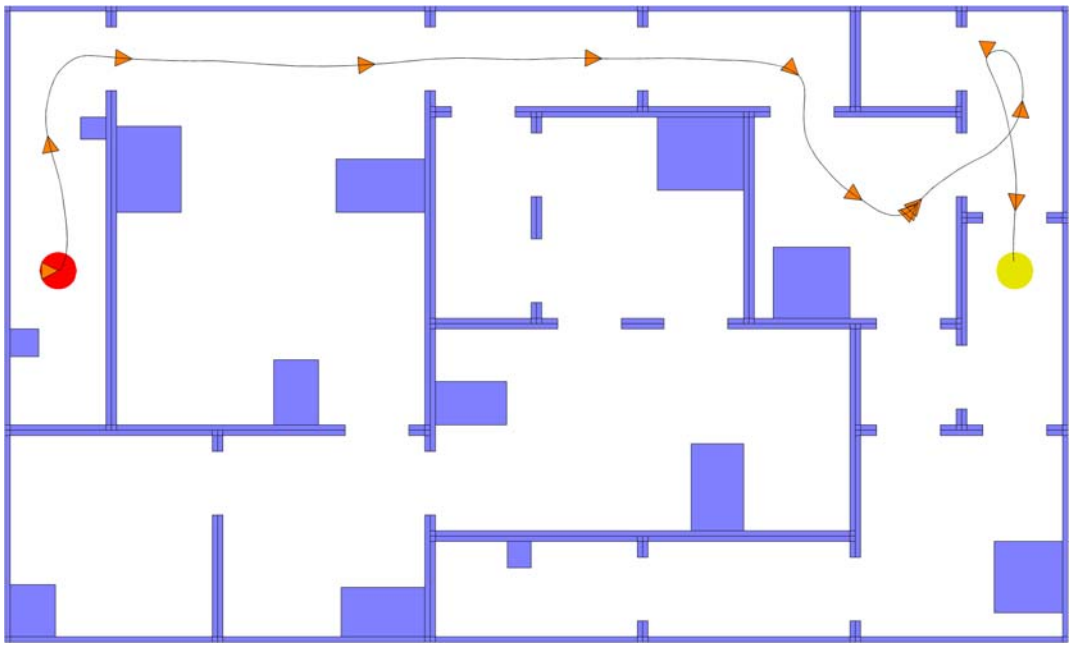

Figure 6.19: The robot's path with W_{v1} = 1.

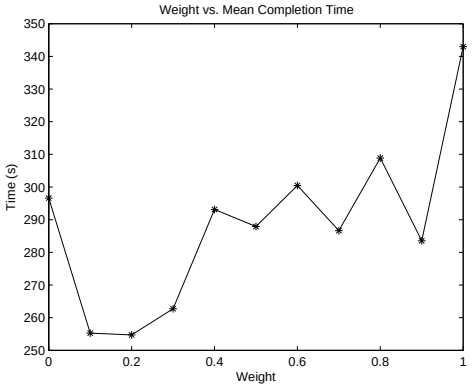

Figure 6.20: Mean completion time for 20 experimental runs per W_{v1} value.

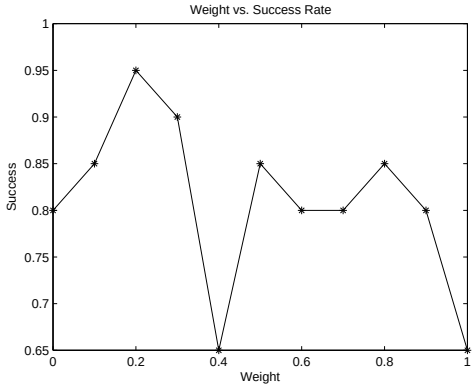

Figure 6.21: Mean success rate vs. W_{v1}.

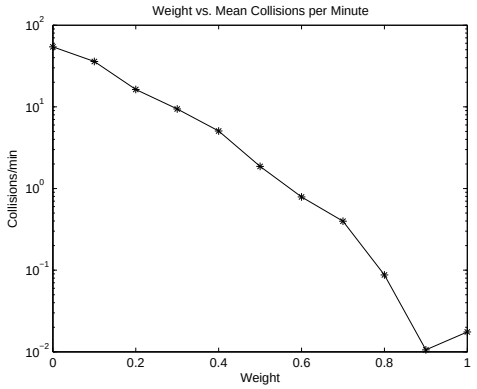

Figure 6.22: Mean collisions per minute vs. W_{v1}.

A slight increase in completion time is observed as W_{v1} increases (Figure 6.20), as a result of the increased tendency to slow down rather than risk collisions. The fluctuations in success rate (Figure 6.21) can largely be attributed to the highly stochastic behaviour resulting from the wander function, but a slight overall decrease is apparent. However, the decrease in the number of collisions by several orders of magnitude (Figure 6.22) indicates that the safety improvements of the velocity controller's avoidance function outweigh the minor cost to performance.

Second, W_{v3} is varied, while W_{v1} = 0.99 (to maximise safety). With the directional controller present, the velocity controller's goal seeking function has a very different effect on performance than that obtained with the velocity controller functioning in isolation (Figure 6.5). Direction inputs are no longer likely to be pointing towards obstacles (due to the influence of the directional controller's avoidance function), so the robot rarely becomes obstructed by them. Increasing W_{v3} actually results in a

general improvement to completion time (Figure 6.23) and success rate (Figure 6.24), rather than having an adverse effect on performance.

A better combination of performance and safety can be achieved by the velocity controller and directional controller acting in unison than either approach can produce independently. Although there is some crossover between the two layers, the directional controller provides superior local goal-directed behaviour, while the velocity controller produces smoother motion and superior obstacle avoidance.

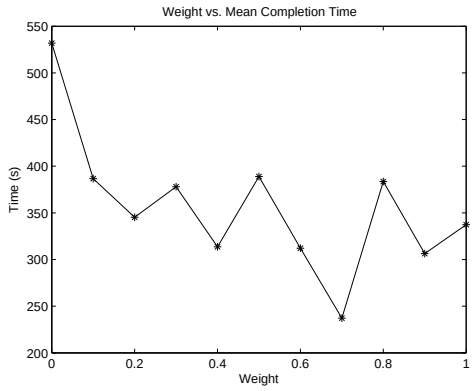

Figure 6.23: Mean completion time for 20 experimental runs per W_{v3} value.

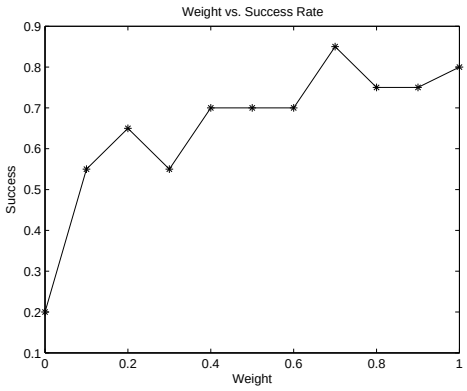

Figure 6.24: Mean success rate vs. W_{v3}.

6.4 Hybrid Reactive/Deliberative Control

The deliberative controller is enabled by setting the path following weight $W_{\theta 3}$ to a non-zero number. The new constant parameter values are shown in Table 6.3, in addition to the values previously shown in Table 6.1 and Table 6.2.

Initially, the directional controller's weights associated with reactive navigation are disabled ($W_{\theta 4} = 0$, $W_{\theta 5} = 0$, $W_{\theta 6} = 0$), while the path following weight $W_{\theta 3}$ is iteratively increased. This experiment is performed first with pregenerated occupancy grid maps of the environments, and then it is repeated without them. In the latter case, the robot constructs its occupancy grid map autonomously as it explores the environment.

TABLE 6.3: PARAMETERS – HYBRID CONTROLLER EXPERIMENTS

Symbol	Name	Value	Justification
W_{p1}	Occupancy weight	1	This is the primary contributor to path planning, so it is generally kept at the maximum value.
W_{p2}	Danger weight	0	Not relevant in the environments utilised for these experiments.
W_{p3}	Exploration weight	0	Not tested in these experiments.
$(\varepsilon_1,\varepsilon_2)$	Occupancy update factors	(0.01, 0.01)	A relatively slow update rate that ensures stability. Increase and decrease rates are equal.
ε_3	Exploration update factor	0.01	Ensures that a node has been well-explored before it reaches a value of 1.
d_l	Look-ahead distance	0.7 m	The length of two horizontally or vertically aligned map nodes. Provides high incentive to adhere to planned path, without being so high as to induce oscillations.
r_f	Map filter radius	0.7 m	Set to twice the robot's radius, or two map nodes. Ensures that paths do not pass too close to walls, without eliminating too much detail.
B	Planning base	10000	Yields a large separation between high and low cost values, but maintains reasonable consistency between similar values.
t_r	Replan threshold time	5 s	Provides relatively rapid response to map updates without being computationally inefficient.

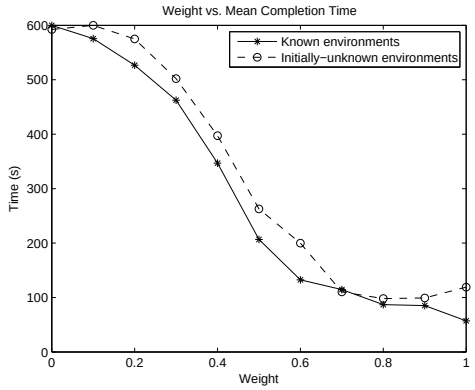

Figure 6.25: Mean completion time for 20 experimental runs per $W_{\theta 3}$ value.

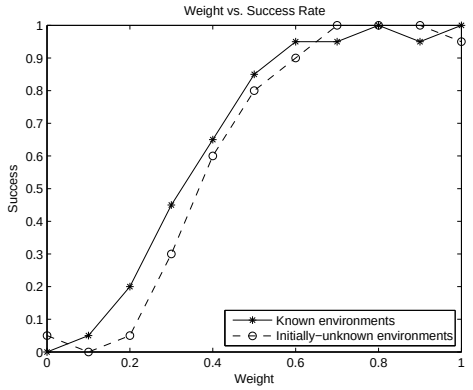

Figure 6.26: Success rate vs. $W_{\theta 3}$.

Regardless of whether the robot has prior knowledge of the environment, the addition of deliberative planning has a positive effect on performance, as shown in Figure 6.25 and Figure 6.26. When path-following is strongly enabled, the robot achieves a 95-100% success rate under both conditions. The best mean completion time is 57 s for known environments and 98 s for initially-unknown environments, both significant improvements over the best results obtained by the reactive system.

Figure 6.27 shows an optimal path that can be achieved in a known environment. When the environment is initially unknown, the robot's path is typically less optimal (Figure 6.28), but nevertheless superior to those achieved by reactive control alone. There are rare problems that can occur, however.

The robot occasionally encounters an environmental configuration where it repeatedly fails to turn into a doorway (Figure 6.29). When the doorway opening is detected by the directional controller, the robot passes the opening too quickly to turn into it. This is caused by the combination of high speeds and high safety margins employed in this experiment. Lower speeds would allow the robot to slow down sufficiently to turn towards the openings, while lower safety margins would enable the directional controller to detect them sooner. However, such changes would also impact performance and safety, as will be demonstrated in Chapter 7.

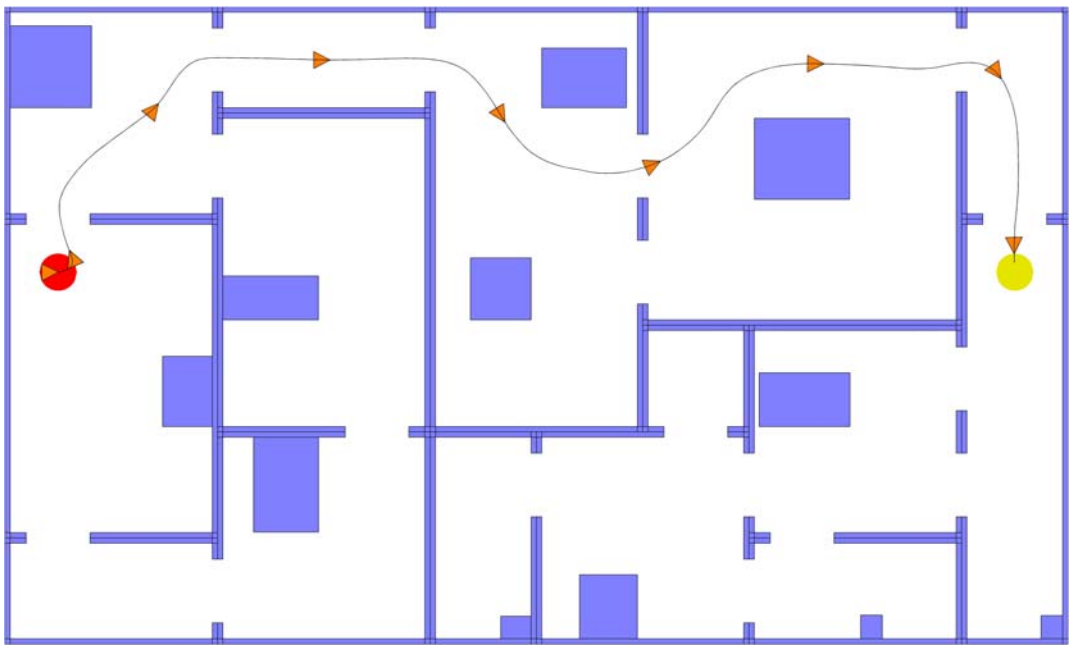

Figure 6.27: A known environment traversed with $W_{\theta 3}$ = 0.8.

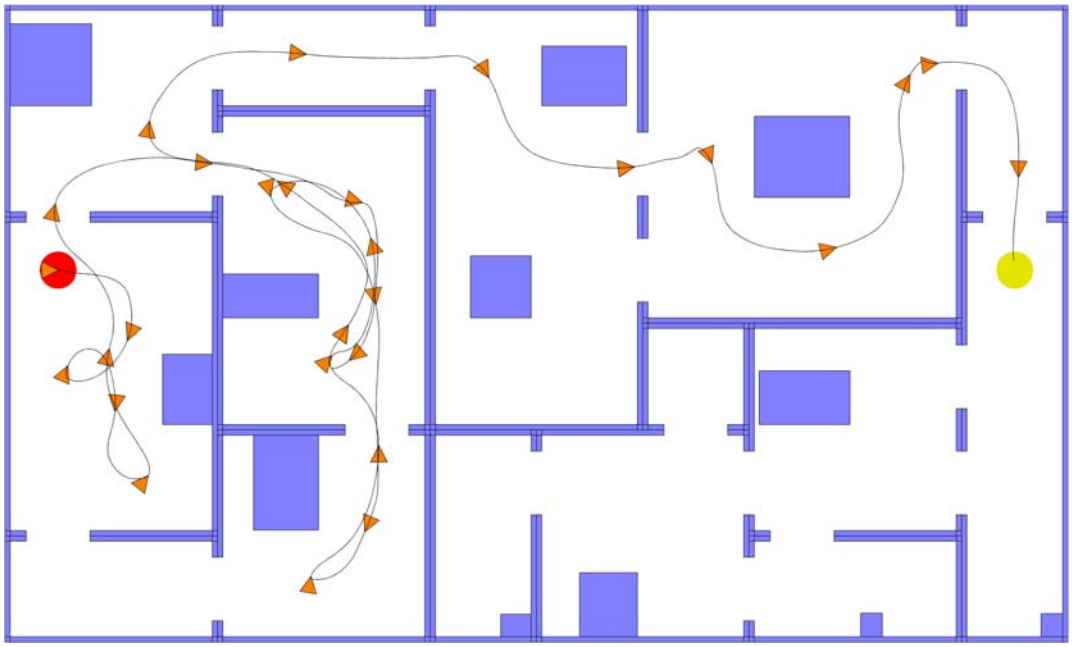

Figure 6.28: An initially-unknown environment traversed with $W_{\theta 3}$ = 0.8.

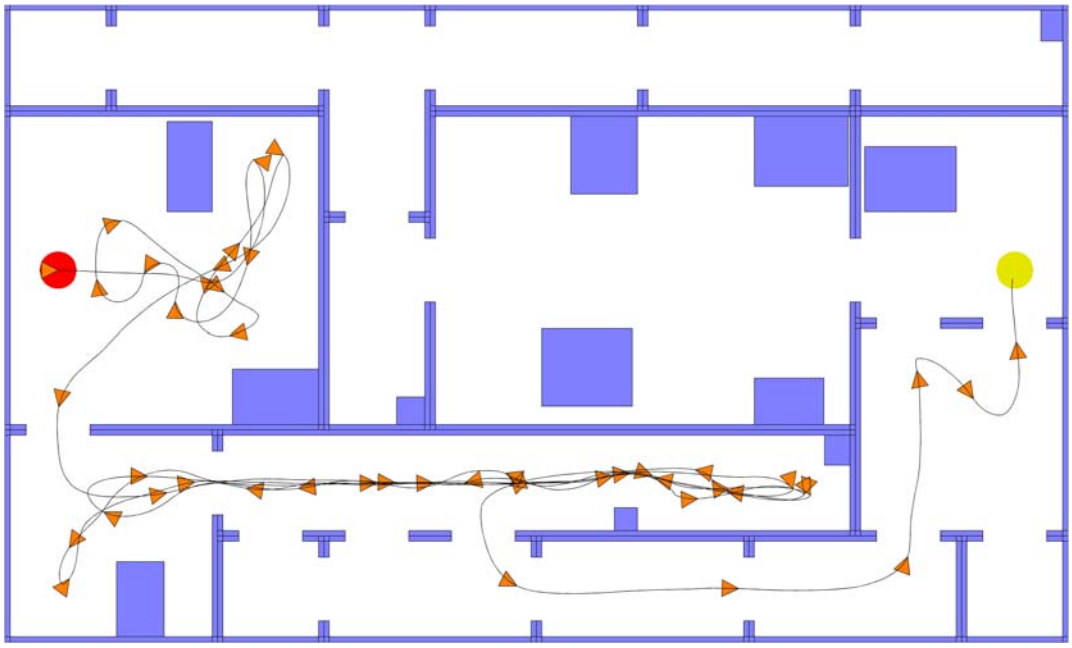

Figure 6.29: An initially-unknown environment traversed with $W_{\theta 3}$ = 0.7.

Another problem that occurs even more rarely is that the robot stops moving if it encounters certain obstacle configurations from certain angles. This is a result of a conflict between the directional and velocity controllers. The directional controller instructs the robot to move in a particular direction, but the velocity controller determines that the direction is unsafe, and due to a confluence of competing goals, it

chooses to neither move, nor turn. Because movement ceases, the only changes to sensor data are due to noise, so the obstructed state persists unless the noise provides sufficient incentive to overcome the obstruction.

In the next experiment, weights associated with reactive navigation are varied, while $W_{\theta 3} = 0.75$ (a value that results in satisfactory performance, according to the results of the previous experiment). The reactive weights are controlled by a multiplier M, which is iteratively increased from 0 to 1 ($W_{\theta 4} = 0.5M$, $W_{\theta 5} = 0.5M$, $W_{\theta 6} = 0.75M$). When $M = 1$, the weights result in satisfactory reactive performance, according to previous results. However, when these weights are strongly enabled in conjunction with path following, they result in a decrease in performance. The completion time shows a general, but erratic, increase as M increases. In the majority of cases, the direction selected by the path following function is superior to the ones chosen by the reactive control functions, so there is little reason to enable reactive goal-directed behaviours when the deliberative planner is functioning correctly.

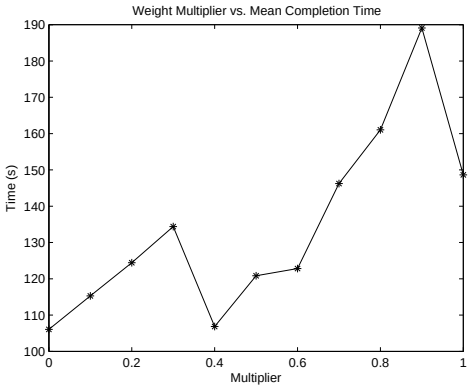

Figure 6.30: Mean completion time for 20 experimental runs per *M* value.

Reactive navigation is not necessarily obsolete because there may be situations where it provides superior performance to hybrid reactive/deliberative navigation. However, these results suggest that it is better to enable the reactive navigation functions only when required, rather than for them to be operating simultaneously with path planning.

6.5 Summary

Some basic navigation experiments have been conducted in a single set of procedurally-generated indoor environments that demonstrate the capabilities of the three main planning and control layers. These experiments involved varying certain weights to partially or fully disable or enable specific components. Thus, the quantitative effects of these components on performance were measured under varying degrees of activation.

While certain weights were varied, the majority of the robot's parameters were maintained at constant values intended (but not proven) to provide satisfactory performance. In the next chapter, different parameter configurations will be tested in a wider range of environments. This is necessary in order to categorise parameters by their effects on performance, so that they can be linked to the robot's drives, and ultimately modulated by the model of robotic affect.

7 Parameter Characterisation

Chapter 6 demonstrated the basic functionality of the robot's different control layers. By changing various control weights, the level of influence of certain functions was altered, and their effect on performance measured. Next, the behavioural changes that arise from varying other planning and control parameters will be quantitatively analysed. This allows the parameters to be sorted into different categories, based on their primary influence on performance. Each category corresponds to a different drive, which controls the robot's bias towards or against a particular 'mode of behaviour'.

Experiments that require equivalent conditions share common sets of environments, to ensure consistency between them. However, due to the diverse requirements of the experiments, there are a number of environment sets, each with different properties. For example, one set of static environments has wide doorways and narrow walls (set B, utilised throughout Chapter 6), while another set has random door and wall sizes. The environment sets are shown in Table 7.1.

TABLE 7.1: ENVIRONMENT SETS

Environment set	**Wall/door sizes**	**Dynamic obstacles**	**Undetectable obstacles**
A	None	No	No
B	Constant	No	No
C	Random	No	No
D	Constant	Yes	No
E	Random	Yes	No
F	Constant	No	Yes
G	Random	No	Yes

Default parameter values obtained from Chapter 6 are shown in Table 7.2. Except where otherwise stated, this is the parameter configuration employed throughout this chapter. While some of the default values are sub-optimal according to results presented in this chapter, they are functional and serve as an adequate starting-point. Typically, a small number of parameters are varied to test their performance characteristics, while the remainder are maintained at these constant values.

TABLE 7.2: DEFAULT PARAMETER VALUES

Parameter	Value	Parameter	Value
W_{v1}	0.99	(n_v,n_ω)	(9, 9)
W_{v2}	0	n_o	5
W_{v3}	0.75	(t_1,t_2)	(0.1 s, 1 s)
W_{v4}	0.9	t_3	1 s
W_{v5}	0.5	r_o	0.35 m
$W_{\theta1}$	0.99	$d_{v(\max)}$	1 m
$W_{\theta2}$	0	β	0.4
$W_{\theta3}$	0.75	n_θ	50
$W_{\theta4}$	0	$d_{o(\max)}$	3 m
$W_{\theta5}$	0	s	0.1
$W_{\theta6}$	0	$(\varepsilon_1,\varepsilon_2)$	(0.01, 0.01)
W_{p1}	1	ε_3	0.01
W_{p2}	0	d_l	0.7 m
W_{p3}	0	r_f	0.7 m
(v_L,ω_L)	(1 m/s, 0.2π rad/s)	B	10000
$(a_{\max},\alpha_{\max})$	(0.5 m/s^2, π rad/s^2)	t_r	5 s
$(a_{\min},\alpha_{\min})$	(-0.5 m/s^2, -π rad/s^2)		

7.1 Safety Parameters

This category includes safety margins and other parameters that directly influence the bias between safety and goal completion. They are important candidates for adaptive modulation, because certain extreme environments and situations are expected to require the robot to disregard its safety in order to achieve its goals, but under more benign conditions the robot can be successful without compromising its safety standards.

These parameters are tested in environment set C, which contains doorways and walls with random widths. Some doorways are very narrow, with the minimum width being 80 cm, only 10 cm wider than the robot itself. Under these conditions, the default safety parameters limit the robot's ability to reach the goal point, so reducing their bias towards safety can improve performance, at the cost of increasing the collision count.

7.1.1 Obstacle Radius

Both the velocity controller and directional controller enlarge obstacles by a radius r_o, either implicitly or explicitly. By default, this is set to the robot's radius, or 0.35 m. If the robot's sensors were 100% accurate and covered 100% of the environment, this would be sufficient to effectively eliminate all collisions with stationary obstacles. In practice, collisions can still occur due to sensor and control limitations, but they are very rare in static environments. However, if this radius is employed in environment set C, the enlargement of the sides of narrow doorways tends to block the robot's path through them. This impedes its progress towards the goal point, or prevents it altogether (Figure 7.1). Reducing r_o increases the size of narrow doorway openings, allowing the robot to more easily traverse them (Figure 7.2).

To test the overall effects of different r_o values in these environments, r_o is incrementally varied from 0.1 m to 0.4 m, and the resulting performance characteristics are recorded. The completion times (Figure 7.3) and success rates (Figure 7.4) are significantly improved for lower r_o values. A 95-100% success rate is achieved when $r_o < 0.25$ m. This improvement does come at a cost to safety, resulting in an increasing collision rate (Figure 7.5) that exceeds 0.5 collisions per minute when $r_o < 0.2$ m.

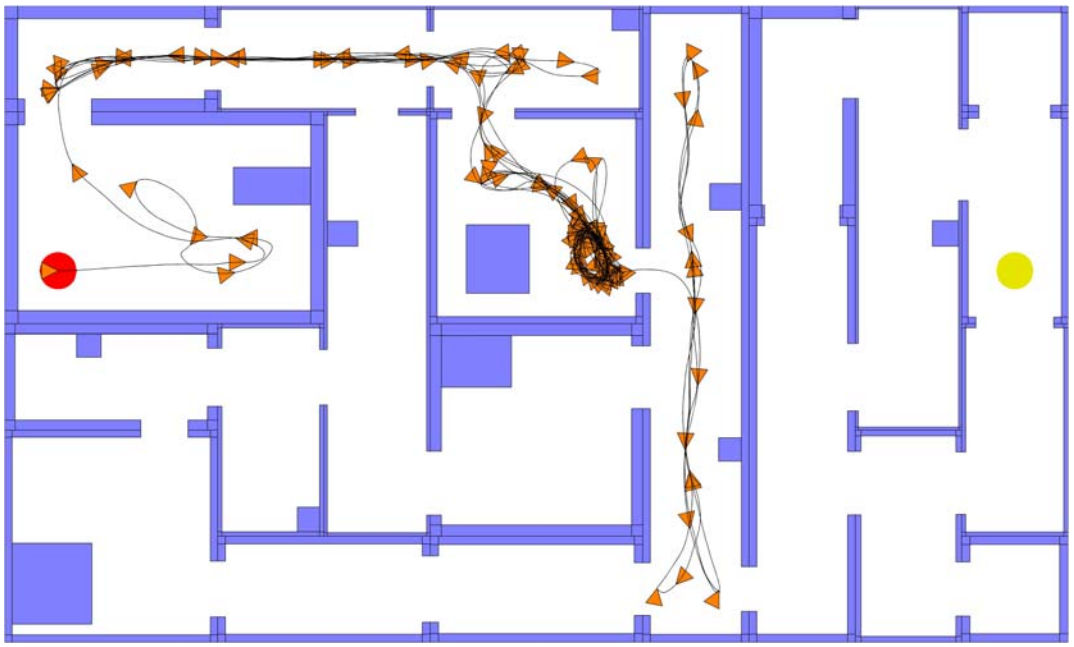

Figure 7.1: Robot's path with r_o = 0.35 m.

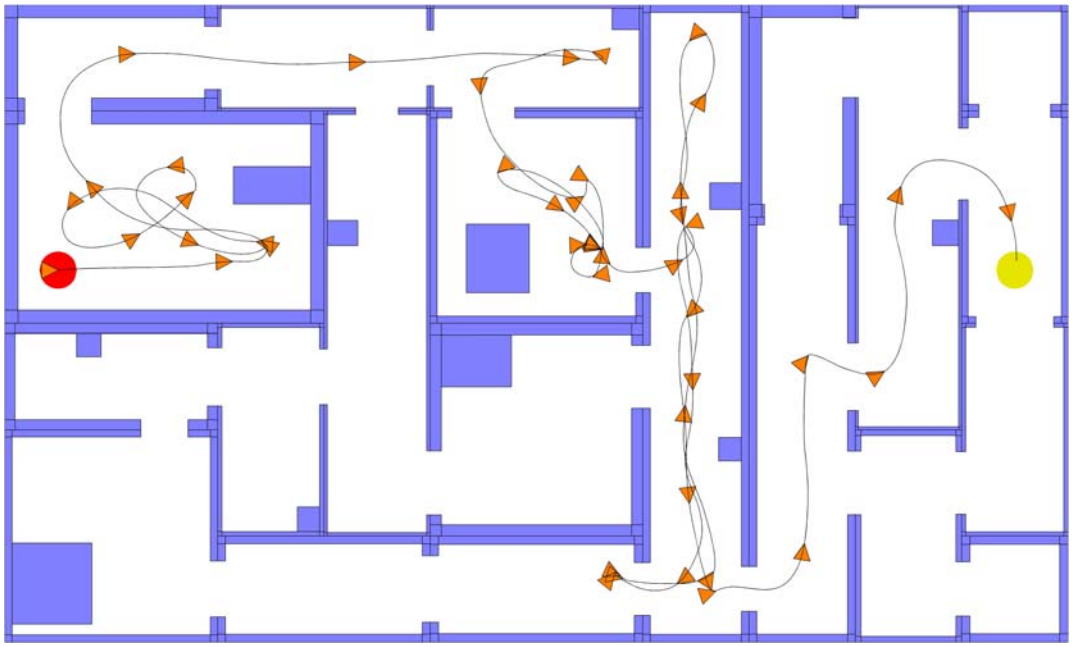

Figure 7.2: Robot's path with r_o = 0.25 m.

These results indicate that there are no r_o values that optimise both safety and performance in environments with narrow doorways. The robot must make a tradeoff between the two requirements. However, the results may be improved if r_o is maintained at high values during normal operation, and only lowered when the robot needs to pass through a narrow doorway.

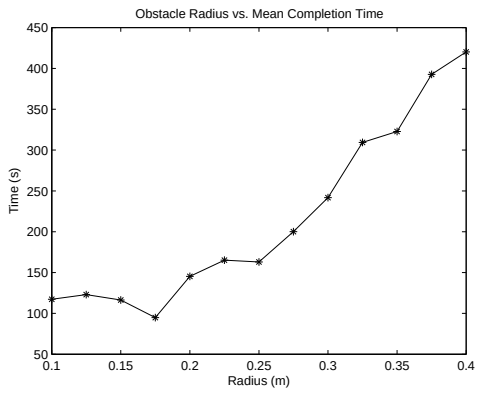

Figure 7.3: Mean completion time for 20 samples per r_o value.

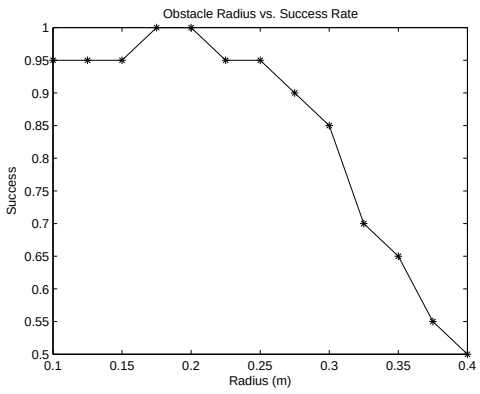

Figure 7.4: Mean success rate vs. r_o.

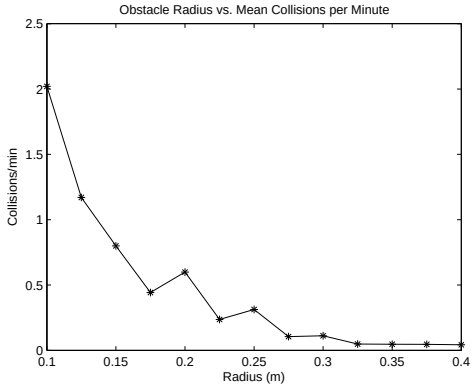

Figure 7.5: Mean collisions per minute vs. r_o.

7.1.2 Smoothing Factor

The smoothing factor s determines the degree of filtering applied to the obstacle distance vector field by the directional controller. It represents the portion of the vector field smoothed in either direction from a given vector, so the percentage smoothed is actually double the value of s. High s values increase the range of influence of obstacles over nearby avoidance vectors, encouraging the robot to give obstacles a wider berth.

To establish a reasonable performance baseline, the obstacle radius r_o is set to 0.25 m (rather than its default 0.35 m) during this experiment. The smoothing factor s is incrementally increased to measure its effects. In a similar manner to increasing the obstacle radius, this should improve safety at a cost to performance. However, in practice, both performance and safety are improved. When s is low, the robot tends to become obstructed by certain obstacle configurations such as doorway corners (Figure 7.6). Large values help prevent this from happening, resulting in smoother, safer and faster progress towards the goal (Figure 7.7).

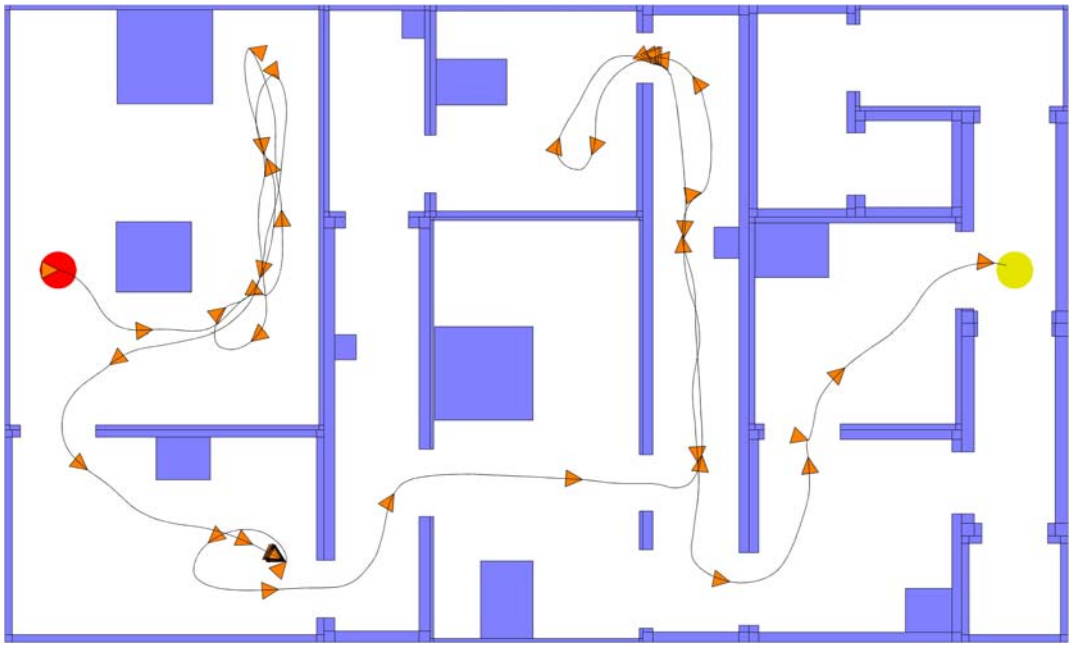

Figure 7.6: Robot's path with s = 0.

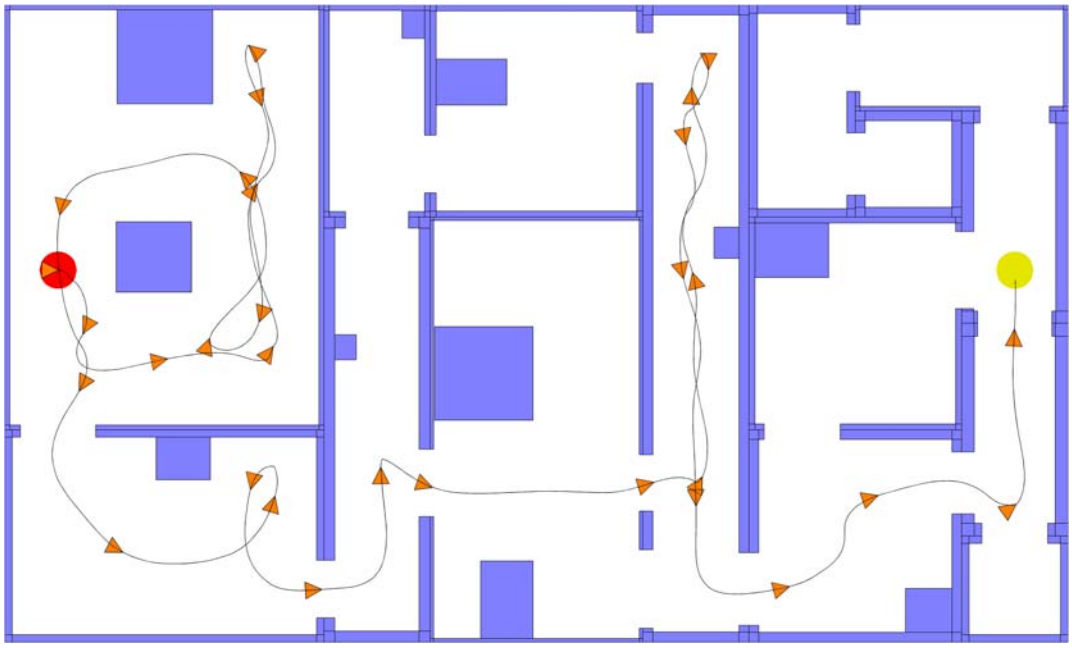

Figure 7.7: Robot's path with s = 0.5.

The mean completion time (Figure 7.8) shows no consistent trend until around s = 0.3, but it begins to decrease after that point. This trend continues until s reaches its maximum of 0.5 (which represents 100% coverage of the vector field). The number of collisions per minute (Figure 7.9) decreases until s = 0.2, and remains largely static thereafter.

The reason for the improvement to performance is likely related to the cause of the robot's obstructed states that occur in these environments. As previously mentioned in Chapter 6, occasionally a conflict occurs between the two reactive control layers when the directional controller instructs the velocity controller to move the robot in a direction that it considers unsafe. Various components of the velocity controller's objective function counterbalance each other to produce linear and angular velocities of zero. The resulting obstructed state can persist for some time, because the robot is stationary and cannot acquire new sensor data that might free it from the obstruction.

Increasing the smoothing factor decreases the likelihood of this conflict occurring by encouraging the directional controller to select directions that are further from obstacles. The lower incidence of obstructed states yields a slight, but tangible improvement to performance, counteracting any performance decrease that might arise by reducing the favourability of vectors that pass through doorways.

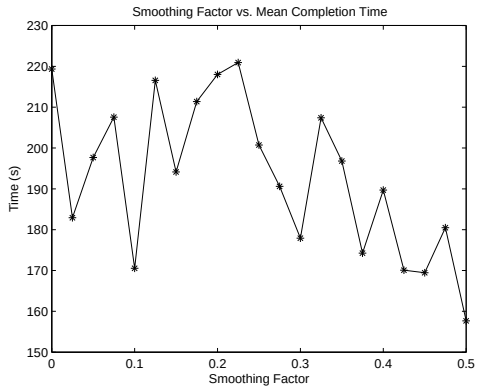

Figure 7.8: Mean completion time for 20 samples per *s* value.

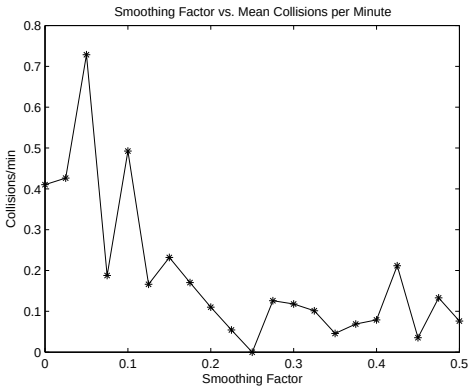

Figure 7.9: Mean collisions per minute vs. *s*.

7.1.3 Avoidance Curvature Time

The avoidance curvature time t_2 controls the size of a circular arc generated by the velocity controller's avoidance function for each velocity couplet. A given curvature is favoured by the avoidance function if it is considered safe for the robot to stay on it for the duration t_2.

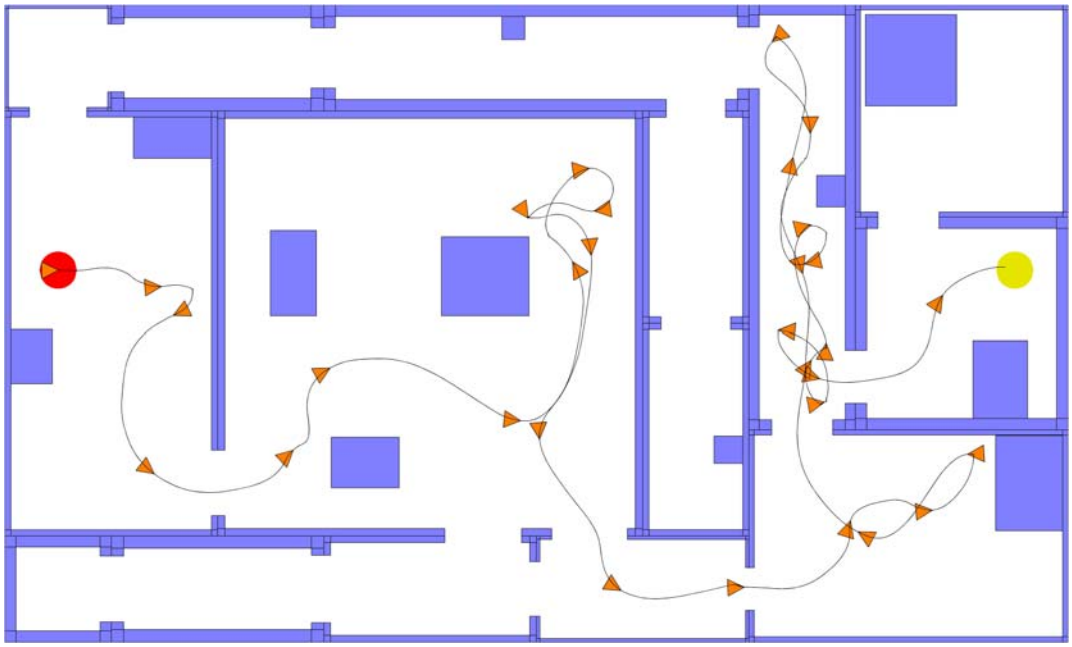

Figure 7.10: Robot's path with t_2 = 0.1 s.

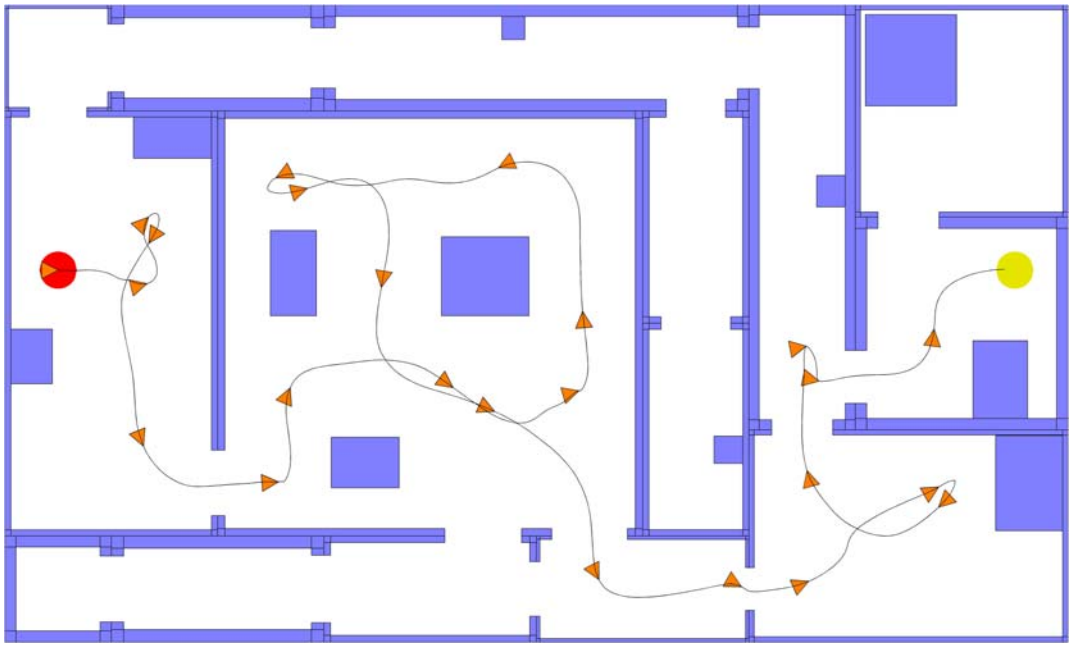

Figure 7.11: Robot's path with t_2 = 1.0 s.

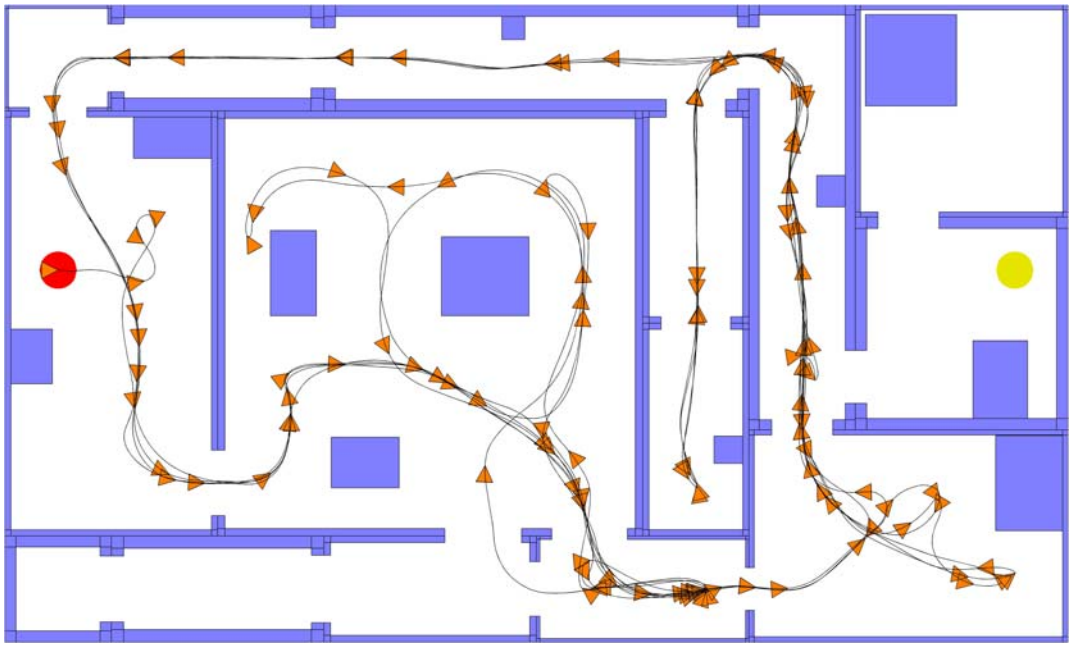

Figure 7.12: Robot's path with t_2 = 2.0 s.

To test the effects of different values of t_2 on performance, a navigation experiment is conducted with r_o = 0.25 m and s = 0.5 (values that result in relatively optimal behaviour in these environments, according to the results of previous experiments). Small values of t_2 mean that only the short term ramifications of selecting a particular curvature are considered by the controller when choosing velocities. This can lead to unnecessary collisions, because the robot fails to slow down and turn away early enough when it detects an obstacle in its path (Figure 7.10). Larger t_2 values cause the robot to respond earlier to perceived obstacles, which can prevent potential collisions (Figure 7.11). However, if t_2 is too large, the robot can become overly conservative in its selection of velocities, preventing it from traversing some doorways (Figure 7.12).

Overall, the mean path time (Figure 7.13) remains steady until around t_2 = 0.9 s, and begins increasing thereafter. Similarly, a near-100% success rate (Figure 7.14) is maintained until around t_2 = 1.1 s, after which it reduces sharply. The collision count (Figure 7.15) shows a different trend, decreasing until around t_2 = 1.1 s, and becoming relatively constant thereafter. There is clearly an optimal region around t_2 = 1.0 s that achieves the benefits of both high performance and high safety. Only small changes within the interval [0.9, 1.1] appear worthwhile, because large deviations from this optimal region adversely influence either performance or safety for no discernable benefit.

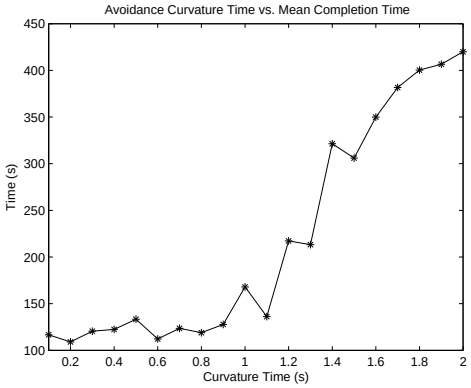

Figure 7.13: Mean completion time for 20 samples per t_2 value.

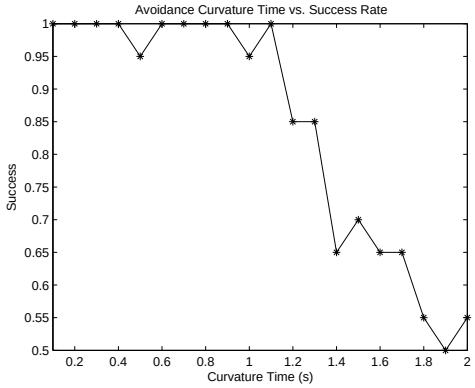

Figure 7.14: Mean success rate vs. t_2.

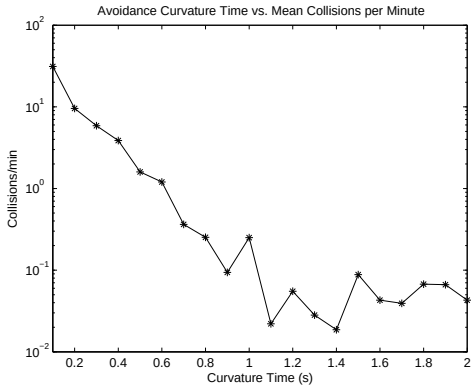

Figure 7.15: Mean collisions per minute vs. t_2.

7.2 Speed Parameters

The speed category includes kinematic and dynamic constraints affecting the velocities contained within the dynamic window. The best way to improve safety in environments containing moving obstacles is to reduce the robot's velocity. Lower velocities afford the robot more time to avoid collisions, and also reduce the potential damage resulting from any collisions that do occur.

7.2.1 Linear Velocity Limit

The robot's global maximum velocity is controlled by the linear velocity limit v_L. This is distinct from the local maximum linear velocity represented in the dynamic window, which constrains the velocities that can be selected in the next control cycle.

The local maximum cannot exceed v_L, but it is primarily a function of the robot's current velocity and acceleration/deceleration limits.

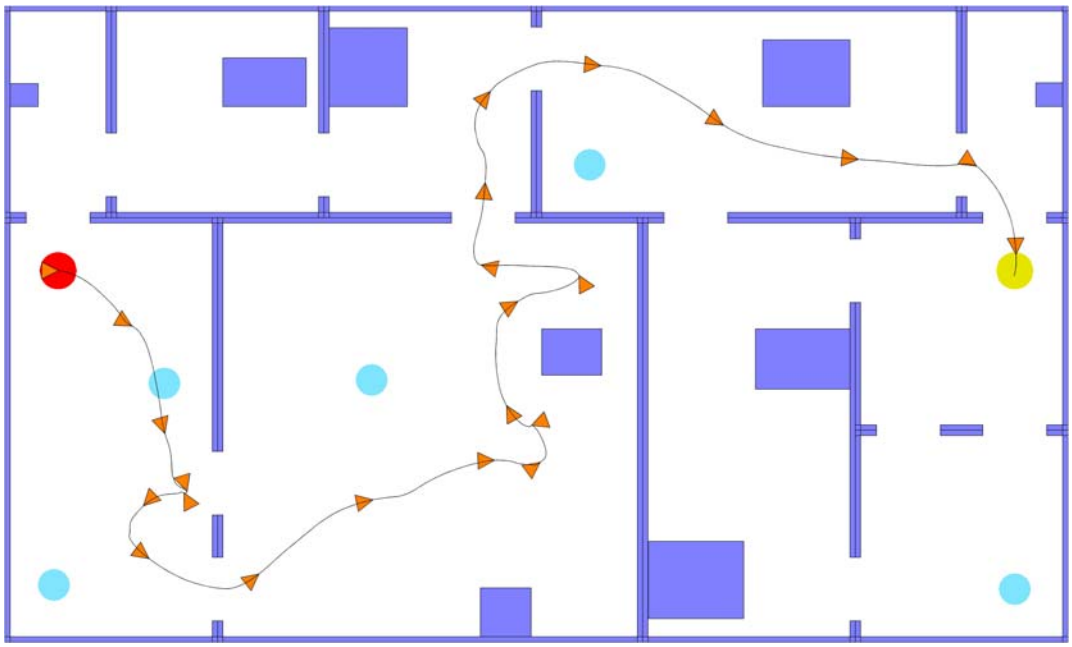

Figure 7.16: A dynamic environment traversed with v_L = 0.6 m/s. The cyan circles represent the initial positions of simulated humans.

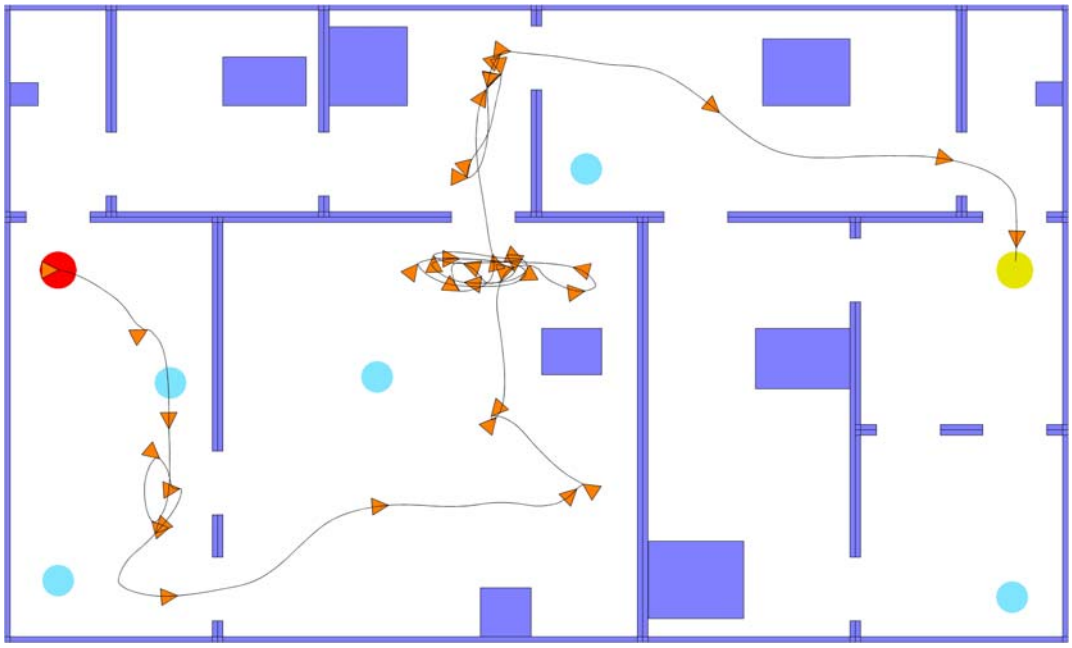

Figure 7.17: A dynamic environment traversed with v_L = 1.2 m/s.

A range of v_L values are tested in two different environment sets: A and C. Environment set D contains simulated humans moving at 0.8 m/s. At 5 second intervals, each human randomly selects a new direction from four options: towards the

robot, away from the robot, clockwise or anticlockwise around it. If a simulated human is moving towards the robot and gets closer than 0.2 m, he/she will stop. However, if the robot is unable to slow down or avoid the human, it will still collide with him/her.

While it would be reasonable to expect that higher velocities should result in the robot reaching its goal sooner, in practice this is not always the case. Sometimes the robot must wait for an area to be mapped before a path can be planned around it, and/or wait for a simulated human to move out of a doorway. Dynamic obstacles can also interfere with the mapping process, resulting in unpredictable occupancy probabilities. Furthermore, higher speeds increase the likelihood that the robot will overshoot a doorway, rather than smoothly turn into it. In the environment shown in Figure 7.16 and Figure 7.17, the robot's path when v_L = 0.6 m/s is shorter than its path when v_L = 1.2 m/s, somewhat compensating for the lower average speed.

Figure 7.18 and Figure 7.19 compare the system's performance and safety over a range of values in both environment sets. The mean completion time (Figure 7.18) initially shows an overall improvement as v_L increases, until around v_L = 0.7 m/s. In the dynamic environments, a gradual increase occurs over the remainder of the range tested, whereas in the static environments, completion time is largely constant over the interval [0.7, 1.2] m/s, and increases sharply thereafter. The rapid increase can likely be attributed to obstructed states, which occur more frequently at higher speeds. It does not appear in the dynamic environments because simulated humans tend to 'push' the robot free from obstructions.

Any performance improvements attributed to higher velocities come at a cost to safety, as indicated by the higher collision counts as v_L increases (Figure 7.19). In the static environments, a very small number of collisions occur over the entire interval, although there appears to be a marginal increase when v_L > 0.8 m/s. The dynamic environment yields a significant increase in collisions when v_L > 0.7 m/s, because the robot is unable to slow down or turn away in time to avoid simulated humans that unexpectedly move into its path. From these results, there appears to be little incentive to set v_L to values greater than 0.7 m/s, as they tend to impede both safety and performance.

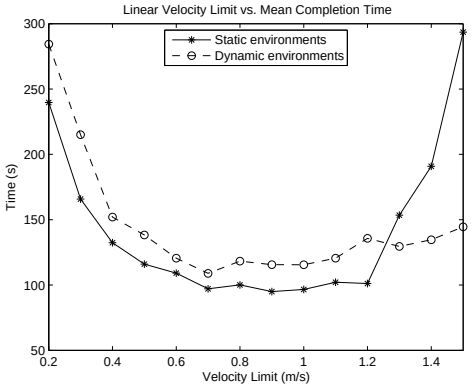

Figure 7.18: Mean completion time for 20 samples per v_L value.

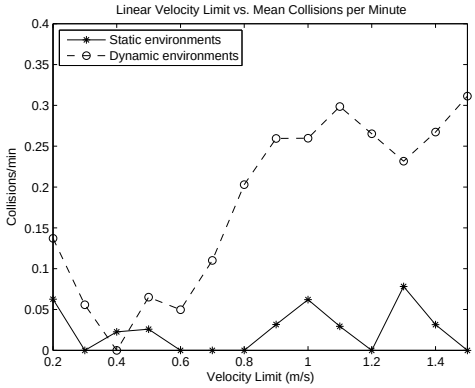

Figure 7.19: Mean collisions per minute vs. v_L.

7.2.2 Linear Deceleration Limit

The linear deceleration limit $a_{\min}$ is a constraint that determines the lowest linear velocity available to the dynamic window, given the robot's current linear velocity. This is distinct from the acceleration limit $a_{\max}$ because some mobile robot drive systems may not accelerate and decelerate at the same rate, and it is not necessarily optimal for the robot to accelerate and decelerate equally. It may be useful to maintain a soft limit that is more severe than a robot drive system's hard limits. Lower magnitudes of $a_{\min}$ (and $a_{\max}$) may reduce wheel slippage and limit the forces exerted on the robot's internal components. However, they are also likely to reduce the robot's responsiveness to environmental dynamics.

As shown in Figure 4.3, the physical robot's deceleration can be as low as -1.6 m/s^2, but only over a small section of the response curve. Throughout the later portions of the curve, it increases to around -0.5 m/s^2. Thus, by default, $a_{\min}$ = -0.5 m/s^2. Different $a_{\min}$ values are tested in environment C, to determine whether it may be worthwhile to modulate $a_{\min}$ in situations where the robot's safety is compromised. Figure 7.20 shows the effects of $a_{\min}$ on the collision count. In general, collisions occur less frequently as $a_{\min}$ is reduced, but improvements to safety are less apparent when $a_{\min}$ < -0.5 m/s^2. These results suggest that while situational reductions to $a_{\min}$ may improve safety, any improvements will be minor.

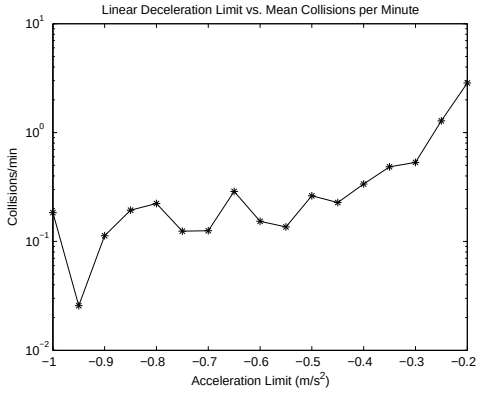

Figure 7.20: Mean collisions per minute vs. a_{min}.

7.3 Efficiency Parameters

Efficiency parameters are those that affect the amount of data that is processed by the controller. Smaller, more granular internal representations yield higher computational efficiency, but lower optimality, which may reduce performance and/or safety. If navigation is a robot's primary function, improving its computational efficiency is likely to lower the energy consumption of its onboard processor(s), increasing battery life. Conversely, if navigation is just one of many processes running in parallel, more computational resources will become available to the other processes.

Experiments with these parameters require a measure of execution speed. This takes the form of execution time ratio, which is the ratio between time represented in the simulation, and the actual time taken to execute the code. The resulting values are highly system dependent, and also dependent on implementation details such as the programming language utilised, the overhead of the simulation, and the attention paid to efficiency-optimisations when writing the code. Thus, the actual values should be viewed as mere indicators, less important than the trends. To maintain consistency between these experiments, they are conducted on the same PC, an Athlon 64 3500+ with 2 GB RAM running Windows XP.

7.3.1 Dynamic Window Sizes

The dynamic window is a two-dimensional rectangular search space of size $n_v \times n_\omega$, where n_v is the number of linear velocities represented in the window, and n_ω is the

number of angular velocities. Varying one or both of these parameters affects the efficiency of the velocity controller by changing the number of objective values that are calculated.

Both parameters are independently varied between 3 and 15 in environment set B. Only odd numbers are utilised, to ensure that there is always a central element representing the robot's current velocity. This is a result of the method utilized to discretise the velocity space – an equal number of velocities are represented above and below the current velocity. Figure 7.21 shows the resulting execution time ratio for each parameter varied. It increases linearly as n_v or n_ω increases, so reducing either of them from their default values clearly improves computational efficiency. This improvement comes at no discernable cost to completion time (Figure 7.22), which remains relatively constant, with only minor fluctuations that can be attributed to random noise. Similarly, the collision count remains close to zero throughout the range of different window sizes.

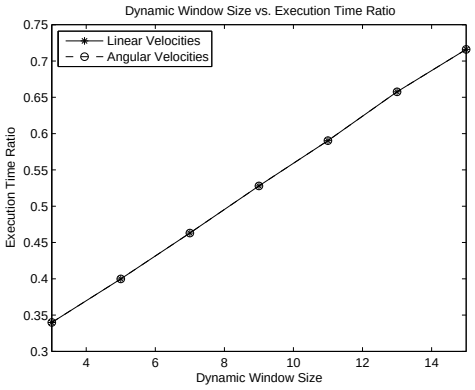

Figure 7.21: Mean execution time ratio for 20 samples per dynamic window size.

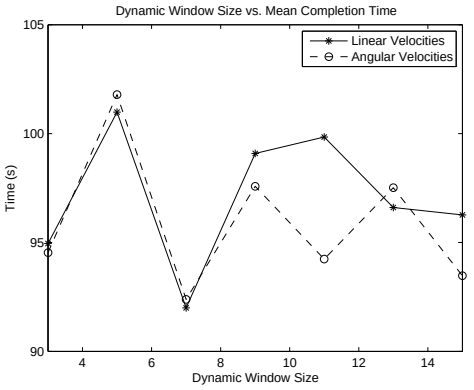

Figure 7.22: Mean completion time vs. dynamic window size.

Regardless of the granularity of the dynamic window, its outer velocities are predominantly the ones selected by the controller. The situations where an inner velocity may be preferable appear to be sufficiently rare or insignificant as to have no detectable influence on performance. It is possible that a controller with less severe kinematic and dynamic constraints than ours would be more likely to select velocities further from the edges of the dynamic window, and that those velocities might allow it to improve performance. But since the different sizes have no discernable effect on safety or performance in our robot, there is little reason to utilise anything other than the most efficient size (3×3). There are no obvious advantages to increasing them over

these minimums, so the dynamic window sizes appear to be poor candidates for adaptive modulation.

7.3.2 Vector Field Size

The directional controller's objective function is applied to a vector field of size n_θ. Computational efficiency should be improved if n_θ is lowered, but this may come at a cost to performance due to the reduced optimality of a more granular representation. For example, fewer direction vectors may yield a reduced likelihood that one of them will pass through narrow doorways, reducing the robot's ability to converge on the goal point.

Vector sizes n_θ between 6 and 60 are tested in environment set B. Based on the results of the previous experiment, the dynamic window size is set to its minimum (and optimal) size of 3×3. The resulting relationship between n_θ and the mean execution time is approximately linear, as shown in Figure 7.23. Changing the vector field size also has a tangible effect on the mean completion time (Figure 7.24), which reduces as n_θ increases, although only when $n_\theta \leq 26$. There is little reason to set n_θ to a value significantly higher than 26, because it no longer provides discernable gains to performance. The collision count remains near zero throughout the entire range, so no relationship between n_θ and safety emerges from this experiment.

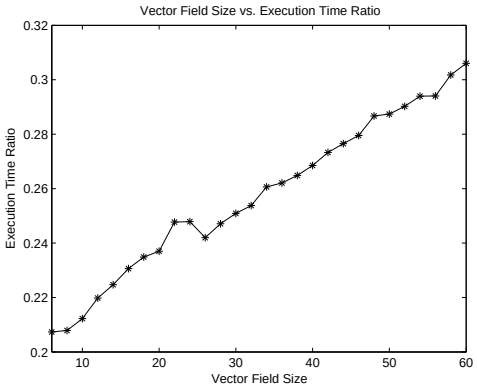

Figure 7.23: Mean execution time ratio for 20 samples per n_θ value.

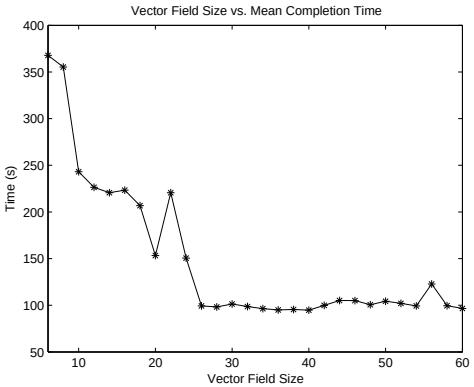

Figure 7.24: Mean completion time vs. n_θ.

7.3.3 Obstacle Buffer Size

Detected obstacle positions are stored in a FIFO buffer for n_o control cycles. In a single control cycle, sensor noise, filtering delays and/or dead zones can make an obstacle appear further away than it actually is. Also, the corners of some objects might be smoothed off if they fall between sensor beams. By considering not only currently detected obstacles, but also previous ones, the limitations of discrete, imperfect sensors can be reduced, potentially improving safety. However, this comes at a cost to computational efficiency, due to the increase in the amount of data processed by each reactive control layer.

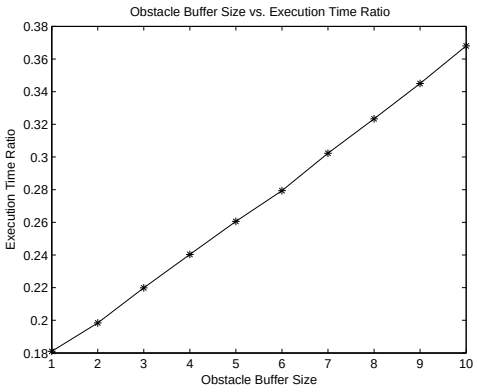

Figure 7.25: Mean execution time ratio for 20 experimental runs per n_o value.

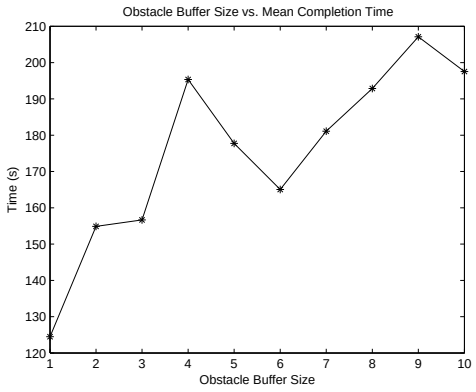

Figure 7.26: Mean completion time vs. n_o.

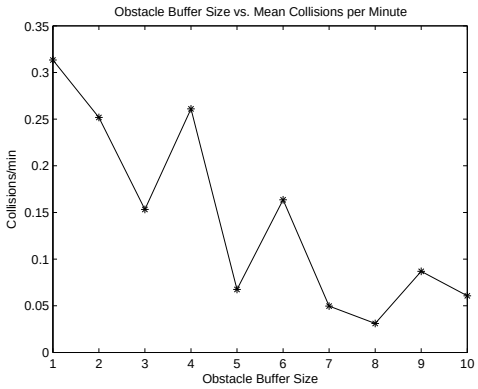

Figure 7.27: Mean collisions per minute vs. n_o.

Due to the emphasis on safety, this parameter is tested in environment set C with obstacle radius r_o = 0.25 m and smoothing factor s = 0.5, values that previously produced relatively optimal behaviour in these environments. Based on the results of the previous two experiments, the dynamic window size is set to 3×3, and the vector

field size is 30. The mean execution time ratio increases linearly with n_o (Figure 7.25). Completion time also shows a general increase (Figure 7.26), although it is subject to large fluctuations. This is countered by an overall decrease in the number of collisions for $n_o \leq 7$ (Figure 7.27). Thus, a tradeoff must be made between safety and efficiency/speed.

7.3.4 Replan Period

The replan period t_r controls how often paths are planned while the robot is navigating from point to point. Decreasing t_r generally has a positive effect on performance, resulting in lower completion times (Figure 7.28). In this implementation, the A* algorithm is applied to a very small map, so it is relatively efficient. In theory, higher t_r values should improve computational efficiency, but in practice efficiency is only measurably impacted when t_r = 0 s (Figure 7.29), as this causes replanning to be performed every control cycle. Thus, a relatively low replan period can be utilised (e.g. t_r = 2 s) without noticeably affecting efficiency. These results are acquired in environment set B with a dynamic window size of 3×3, and a vector field size of 30.

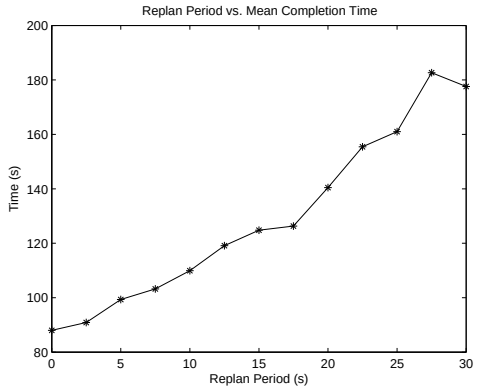

Figure 7.28: Mean completion time for 20 samples per t_r value.

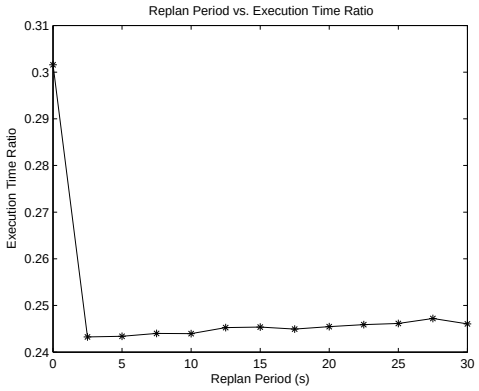

Figure 7.29: Mean execution time ratio vs. t_r.

7.4 Exploration Parameters

This category includes parameters that influence deliberative maps constructed by the robot. In particular, the robot must choose whether to explore the environment and acquire new world knowledge, or to exploit its existing knowledge to plan an optimal path to the goal point. Furthermore, if the robot is exploring a region of the

environment that it has not encountered before, its map update rules may be different than they otherwise would be.

7.4.1 Exploration Weight

The path planner's exploration weight W_{p3} controls the bias between exploration and exploitation. Unmapped areas of the grid map are assigned occupancy values of $p_o = 0.2$.[1] Once the robot begins to explore an area, free space nodes tend towards $p_o = 0$, while occupied nodes tend towards $p_o = 1$. Thus, the robot's default behaviour (when $W_{p3} = 0$) is to favour known free space nodes (with $p_o \approx 0$) over unexplored nodes (with $p_o = 0.2$). When W_{p3} is increased, the cost of known free space nodes begins to meet, and then exceed the cost of unexplored nodes. This results in an increasingly strong bias to explore the environment and update the map, rather than to exploit existing map data to reach the goal quickly.

To measure the effects of W_{p3} on performance, an experiment is conducted where the robot is instructed to travel back and forth between two points in environment set B, for a total of three traversals of each map. When W_{p3} is set to low values, the robot typically finds a satisfactory path to the goal, and follows the same path thereafter (Figure 7.30). Conversely, when exploration is strongly enabled by setting W_{p3} to high values, the robot has a significant disincentive to repeat paths previously travelled, so it covers a larger portion of the map (Figure 7.31).

In general, increasing W_{p3} yields a corresponding increase in the robot's coverage of the environments (Figure 7.32). This trend continues until around $W_{p3} = 0.7$. Coverage increases come at a cost to completion time (Figure 7.33), due to the more indirect paths chosen by the robot as it explores the environments. Completion times are more adversely affected when $W_{p3} > 0.5$. At this point, the difference in cost between occupied and free-space nodes is less than 0.5 in fully-explored areas. The cost factor of occupied nodes therefore becomes only 100 times greater than that of unoccupied nodes (as a comparison, it is 10000 times greater when the cost difference is 1). The robot's path planning capabilities thus become increasingly degraded. When $W_{p3} > 0.7$, the cost factor gain is reduced to less than 15.8, so the resulting planned

[1] The obvious initialisation value is 0.5. However, unless the exploration weight exceeds this value the robot will not favour exploration over following the shortest path, and higher weights degrade the quality of planned paths.

paths often pass through walls and other obstacles, sometimes causing the robot to become permanently obstructed. Ideally, the combined contributions of the exploration and emotion maps (described in Chapter 8) should be kept less than 0.5.

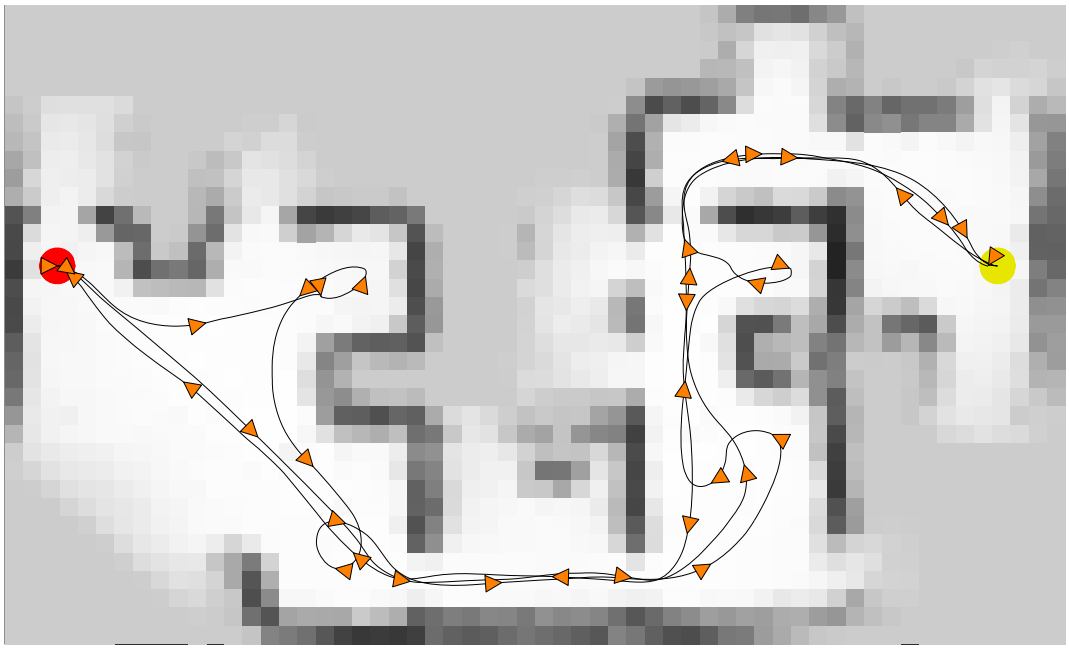

Figure 7.30: Occupancy grid map and robot's path with W_{p3} = 0.

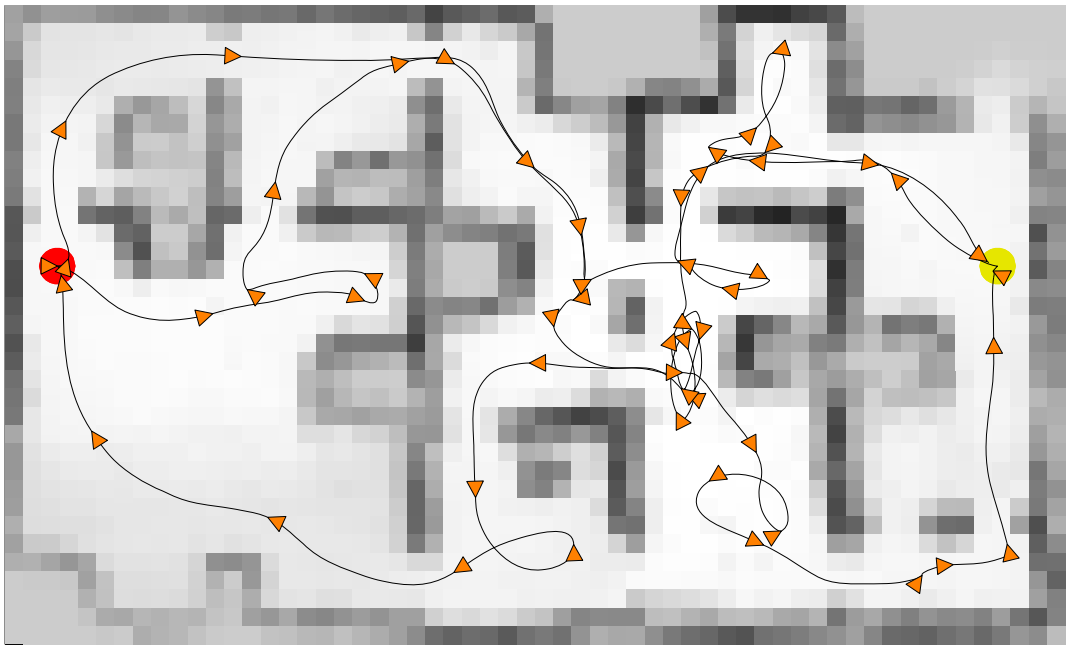

Figure 7.31: Occupancy grid map and robot's path with W_{p3} = 0.5.

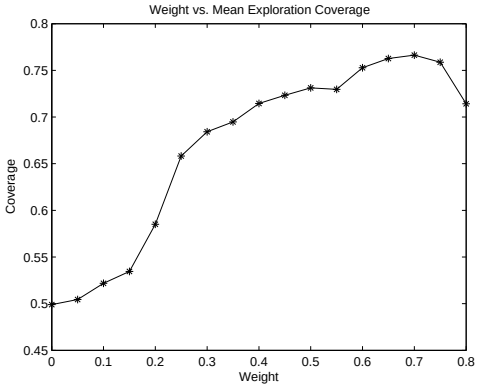

Figure 7.32: Mean exploration coverage for 20 samples per W_{p3} value.

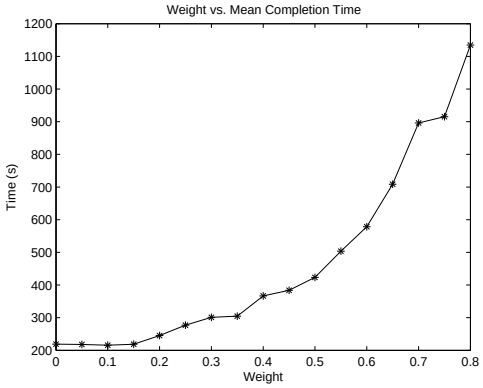

Figure 7.33: Mean completion time vs. W_{p3}.

7.4.2 Map Update Rate

The rate at which the occupancy grid map is updated from instantaneous sensor data is controlled by the update rates ε_1 and ε_2. Reductions in occupancy are affected by ε_1, while increases are affected by ε_2. The ratio between ε_1 and ε_2 influences the robot's implicit bias towards either ensuring that free space nodes are assigned low p_o values, or ensuring that occupied nodes are assigned high p_o values. If $\varepsilon_1 >> \varepsilon_2$, static obstacles may not be clearly represented on the map, resulting in planned paths that pass through walls (Figure 7.34). Conversely, if $\varepsilon_1 << \varepsilon_2$, a dynamic obstacle may persistently appear as an occupied region even after it has moved to a different location (Figure 7.36). The default configuration, $\varepsilon_1 = \varepsilon_2$, typically results in satisfactory paths in dynamic environments (Figure 7.35).

The performance characteristics of different update ratios are measured by keeping $\varepsilon_1 = 0.01$, and varying ε_2. This experiment is conducted in environment sets B and D, representing static and dynamic environments. Both environment types show similar trends. The mean completion time (Figure 7.37) decreases rapidly as ε_2 approaches ε_1, but once ε_2 exceeds ε_1 it remains relatively constant (with minor fluctuations).

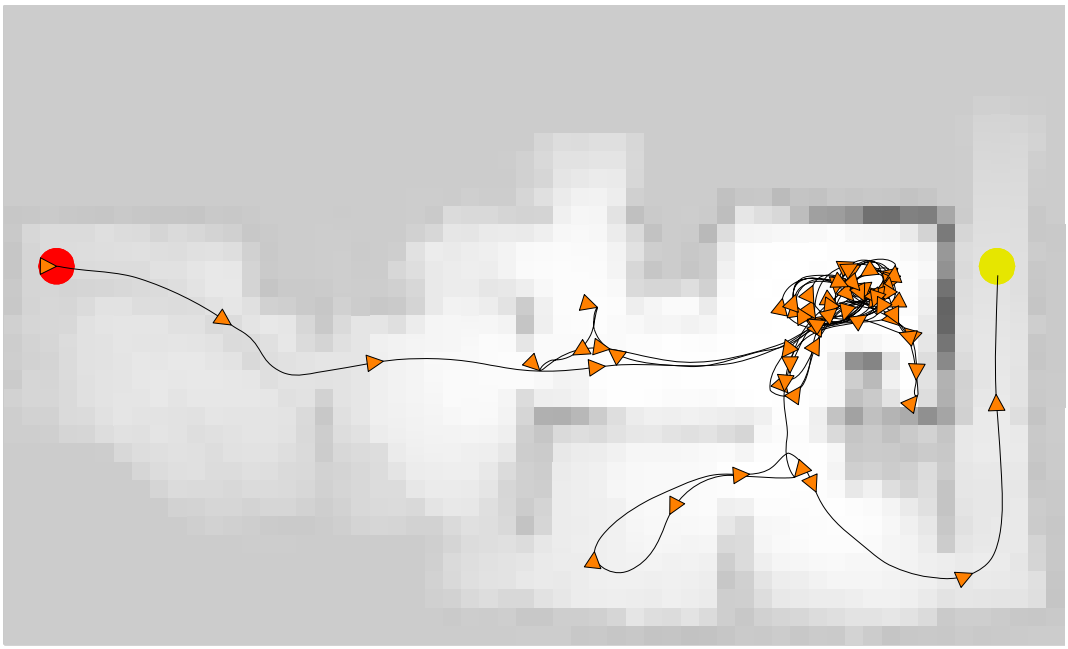

Figure 7.34: Occupancy grid map and robot's path with ε_1 = 0.01, ε_2 = 0.001.

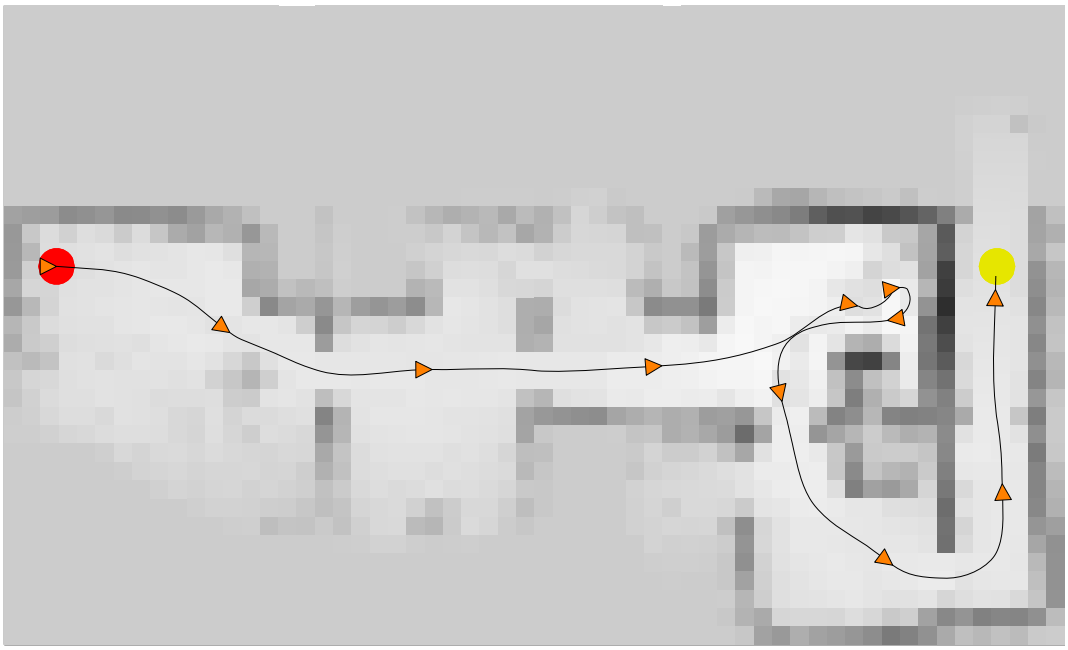

Figure 7.35: Occupancy grid map and robot's path with ε_1 = 0.01, ε_2 = 0.01.

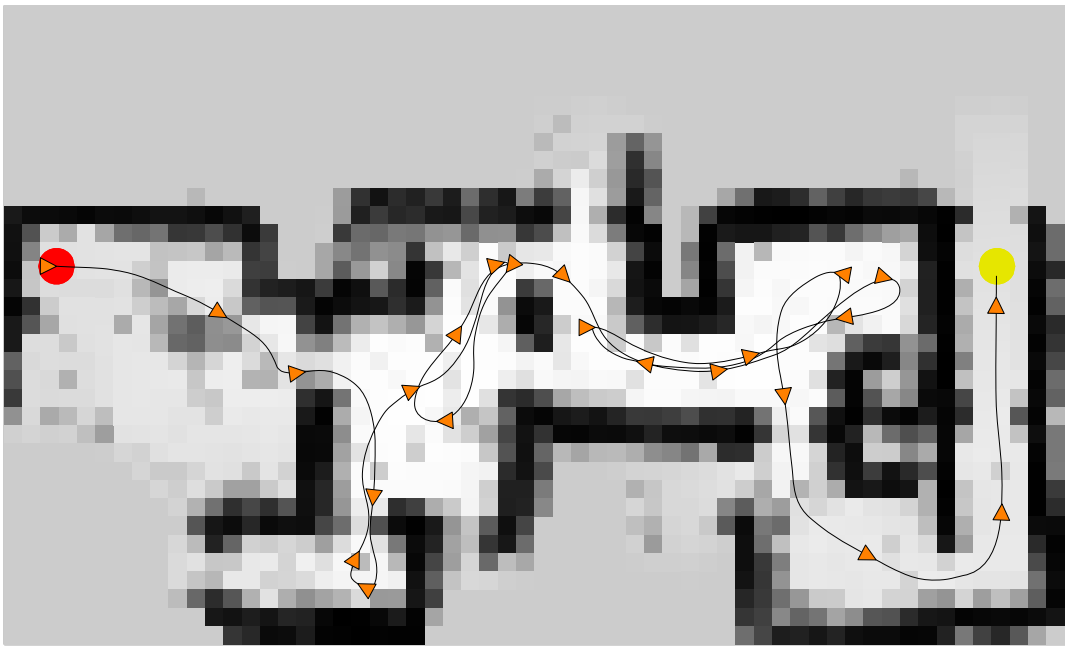

Figure 7.36: Occupancy grid map and robot's path with $\varepsilon_1 = 0.01$, $\varepsilon_2 = 0.1$.

There appears to be little reason to change these values from their defaults if they are to be constant parameters. However, it may be advantageous to increase ε_2 while the robot is exploring a newly discovered area, and reduce it towards its default (0.01) once the area has been explored. Unexplored nodes are initialised to $p_o = 0.2$, so under normal conditions it takes longer for a node to converge on an occupied state ($p_o = 1$) than it does to converge on an unoccupied state ($p_o = 0$). Modulating ε_2 based on exploration requirements may help the robot build its map more rapidly without adversely affecting stability.

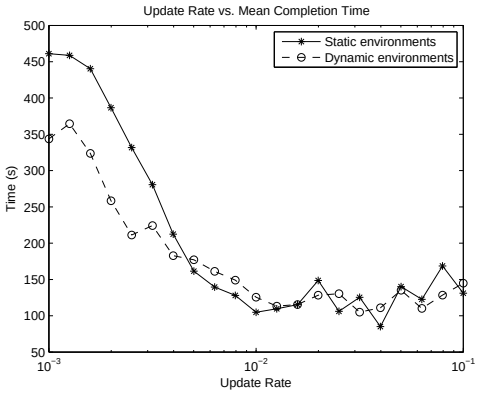

Figure 7.37: Mean completion time for 20 samples per ε_2 value.

7.5 Action Parameters

Action parameters control the bias between reactive and deliberative navigation strategies. Under normal conditions, obstacle avoidance guided only by deliberative path planning produces the best performance and safety. However, there are situations where an emphasis on deliberative navigation could be inferior to a more reactive approach. In particular, while a robot is building its map, planned paths are often sub-optimal, passing through walls and other obstacles.

7.5.1 Directional Control Weights

Some relatively optimal weight configurations were obtained for both reactive and deliberative control in chapter 6. Acceptable reactive performance can be achieved with $W_{\theta 3} = 0$, $W_{\theta 4} = 0.5$, $W_{\theta 5} = 0.5$ and $W_{\theta 6} = 0.75$, while a good configuration for deliberative control is $W_{\theta 3} = 0.5$, $W_{\theta 4} = 0$, $W_{\theta 5} = 0$ and $W_{\theta 6} = 0$. Enabling reactive goal seeking capabilities simultaneously with deliberative path following decreased the robot's performance. Instead, it is envisaged that the weights should be modulated between these two configurations to suit the robot's momentary requirements.

7.5.2 Look-ahead Distance

Another way to shift the bias between reactive and deliberative control is to change the look-ahead distance parameter d_L. Increasing d_L causes the controller to target a point further along the planned path. When d_L exceeds the total path length, the directional controller's path-following function produces the same output as its goal seeking function.

Figure 7.38 shows that the best completion times are normally achieved when d_L = 1.05 m, which is equivalent to the distance between three horizontally or vertically aligned nodes. These results are obtained in environment set B. In situations where the robot's deliberative capabilities are compromised, it may be advantageous to increase d_L, shifting the robot's behaviour towards the reactive end of the spectrum.

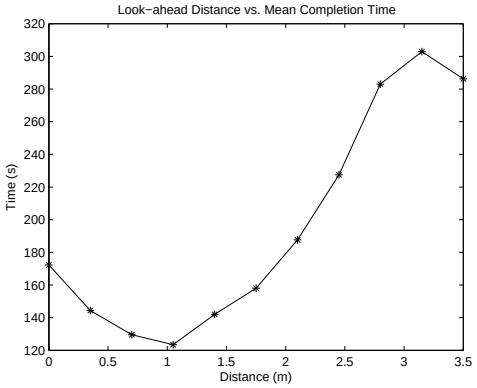

Figure 7.38: Mean completion time for 20 experimental runs per d_L value.

7.6 Introspection Parameters

This category includes parameters that control the influence of internal representations over both reactive obstacle avoidance and deliberative planning. Certain situations may render the robot's sensors unreliable for collision avoidance (e.g. sensor failures or undetectable obstacles). Occupancy and danger grid maps can provide additional information that facilitates collision avoidance, and enables the robot to retain some level of functionality in situations that would otherwise result in failures.

These parameters are tested in environment set F, which includes static environments containing objects that the robot's sensors cannot detect. Real-world objects that may have this property include transparent objects such as windows, and objects that reside outside of the sensors' planes of detection. For testing purposes, the environments are known to the robot (fully mapped) at the onset of each experiment, and map nodes corresponding to undetectable obstacles are tagged as dangerous. The alternative would be to allow the robot to explore and tag the dangerous areas itself (by colliding with the undetectable obstacles), but this would require multiple passes through each environment (and numerous collisions) to achieve equivalent results.

7.6.1 Map-based Avoidance Weights

The strengths of the reactive controllers' map-based avoidance responses are regulated by weights W_{v2} and $W_{\theta 2}$. Increasing these weights increases the robot's

reactive aversion to map nodes that are occupied or tagged as dangerous (either a priori, or as a result of collisions).

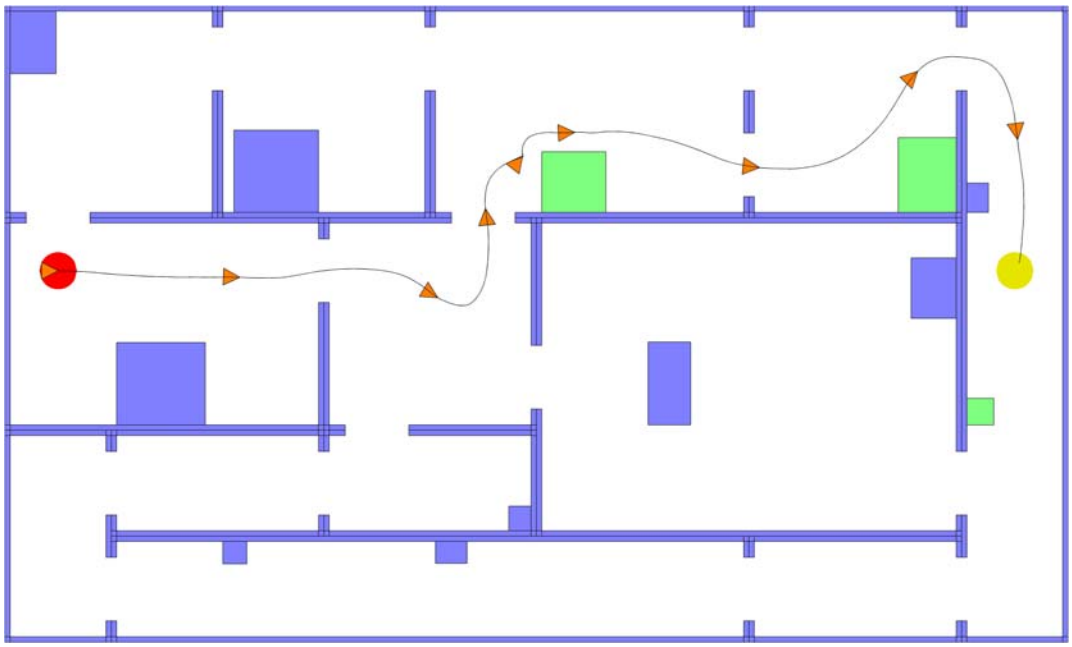

Figure 7.39: Robot's path with W_{v2} = 0 and $W_{\theta 2}$ = 0. The green rectangles represent undetectable objects.

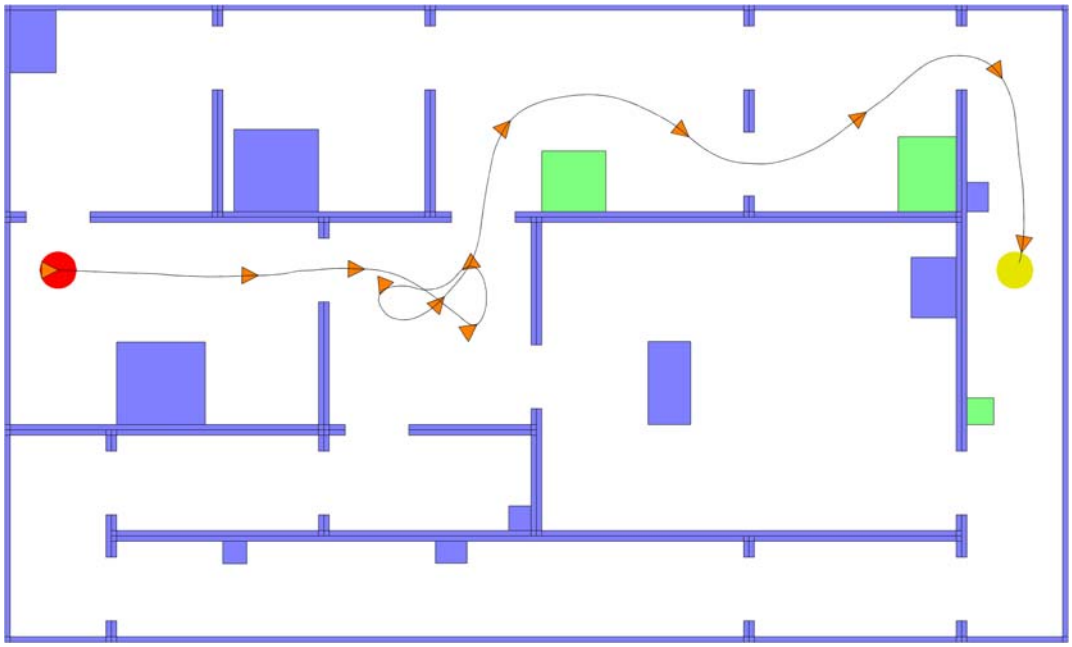

Figure 7.40: Robot's path with W_{v2} = 0.4 and $W_{\theta 2}$ = 0.4.

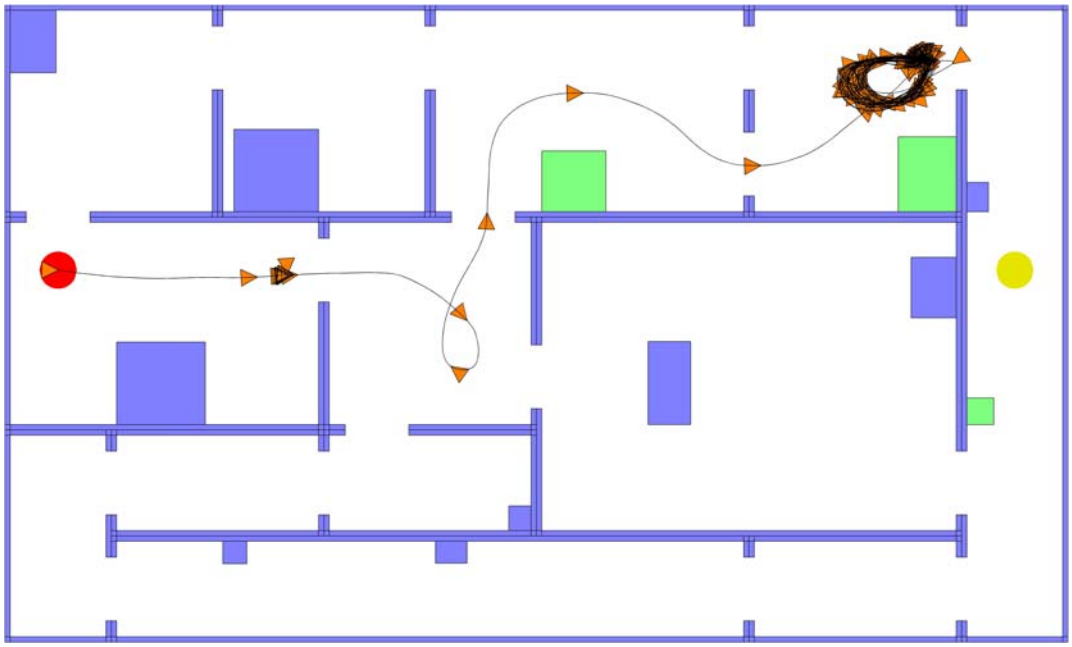

Figure 7.41: Robot's path with W_{v2} = 0.8 and $W_{\theta 2}$ = 0.8.

If these weights are disabled (Figure 7.39), there is nothing to prevent the robot from colliding with the objects it cannot detect. Increasing the weights causes the robot to give these objects a wider berth (Figure 7.40), but it also becomes increasingly conservative in its avoidance of visible objects. The coarse resolution of the grid maps results in situations where the sensor-based obstacles and map-based obstacles are aligned in such a way as to block certain doorways. The problem is exacerbated by higher weights, where the robot's aversion to mapped obstacles approaches that of sensor obstacles (Figure 7.41).

This results in an overall trend where the mean completion time increases with the weight values (Figure 7.42). The mean number of collisions per minute significantly decreases for weights > 0.3, and it approaches zero for weights > 0.5 (Figure 7.43). There is no constant weight configuration that optimises both performance and safety; the robot must select which to favour.

The problems with higher avoidance weights in this experimental scenario could be reduced if the map-based avoidance function only attempted to avoid obstacles on the danger map (rather than a fused occupancy and danger map). However, that would decrease safety in other scenarios (e.g. if sensor failures occur). There also remain situations where such an implementation would result in obstructions (e.g. if the robot collides with the side of a doorway, the point of collision is marked on the danger map

and can cause the same obstructions as occupancy maps). A superior solution is to activate the map-based avoidance weights only when necessary, as will be described further in subsequent chapters.

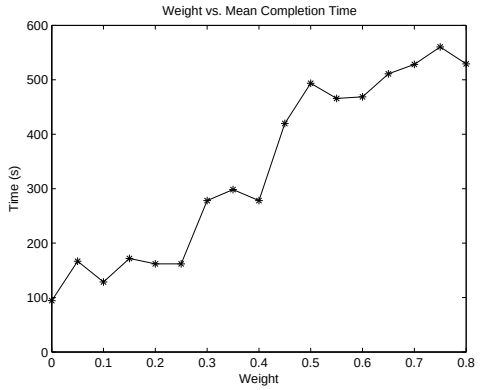

Figure 7.42: Mean completion time for 20 experimental runs per map-based avoidance weight.

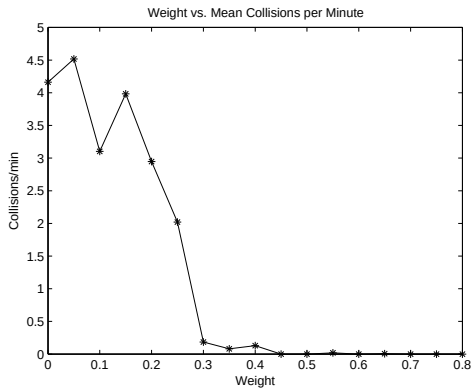

Figure 7.43: Mean collisions per minute vs. map-based avoidance weight.

7.6.2 Danger Weight

The weight W_{p2} biases planned paths to avoid map nodes tagged as dangerous. This is expected to improve safety and prevent the robot from planning paths that are unviable (e.g. through doorways blocked by undetected obstacles). It only affects deliberative planning, not the reactive controllers, so it does not prevent collisions. Nonetheless, the number of collisions should be lowered if the robot follows paths that are further from dangerous locations. Figure 7.44 shows the mean collisions per minute that result from various W_{p2} configurations (with $W_{v2} = 0$ and $W_{\theta 2} = 0$). A significant decrease in collisions occurs between $W_{p2} = 0.1$ and $W_{p2} = 0.2$, but the collision count remains relatively constant thereafter, and never drops below 1 collision per minute.

By default, this weight is disabled, because it is likely counterproductive for the planner to avoid all locations marked as dangerous. For example, if the robot collides with a dynamic obstacle while passing through a doorway, the point of collision would be tagged as dangerous. If W_{p2} is high, this could prevent paths from being planned through that doorway even after the obstacle has moved away. A preferable approach is to activate W_{p2} only when required. This is further explained in Chapter 8.

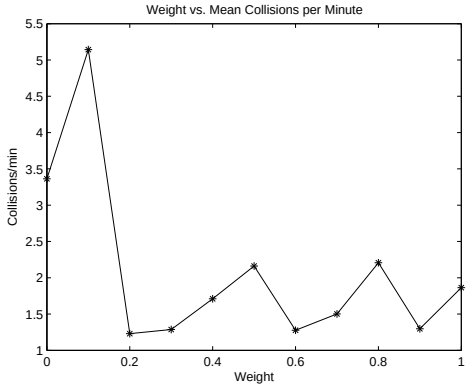

Figure 7.44: Mean collisions per minute for 40 experimental runs per W_{p2} value.

7.7 Summary

The performance characteristics of various planning and control parameters have been quantitatively analysed. Parameters are grouped into different categories based on their behavioural influences. Optimal values and ranges can be derived from these results that will determine whether and how each parameter should be modulated to improve navigational performance and/or safety.

The next chapter will describe the implementation of our robotic affect model, which has the role of modulating these parameters to improve the robot's long-term survival and adaptive capabilities.

8 Robotic Affect Implementation

In Chapters 6-7, some behaviour and performance changes that result from parameter variations were presented. It was shown that certain tradeoffs are necessary between competing requirements such as goal convergence, collision avoidance, computational efficiency and knowledge acquisition. These tradeoffs should take into account many different types of situations and environments that the robot may encounter. No single set of parameters is optimal in every conceivable situation; the robot must sometimes compromise its performance in one state to provide satisfactory performance in another.

Our model of affect is intended to manage some of these tradeoffs by continuously adapting the robot's parameters to suit its perceived internal and external world. The affective system interacts with a distinct cognitive system (represented by the planning and control architecture). As inputs it receives certain sensor data and representation information created by the various control layers, and it generates appropriate parameter configurations and path planning biases in response. As explained in Chapter 3, the model comprises four main components: stimuli, drives, emotions and moods. Now that the underlying planning and control architecture and experimental results have been presented, this chapter describes the implementation details of these components and their various interconnections.

8.1 Affective Stimuli

Stimuli in our system represent events that trigger adaptive control parameter modulations. They are unit interval intensity values that grow or decay in response to the presence or lack of certain eliciting events. Some stimuli (e.g. danger) are continually active and varying in intensity, while others (e.g. error) are dormant during normal operation, only activating in situations that require the robot to change its behaviour.

Raw stimuli S are normalised so that they increase and decrease appropriately within the interval [0, 1]. Normalised stimuli S_{norm} result from linear transformations bounded by upper and lower thresholds $S_{\min}$ and $S_{\max}$:

$$S_{norm} = \begin{cases} 0 & \text{if } S_{raw} < S_{\min} \\ \left(\dfrac{S - S_{\min}}{S_{\max} - S_{\min}} \right) & \text{if } S_{\min} < S_{raw} < S_{\max} \\ 1 & \text{otherwise} \end{cases} \tag{8.1}$$

Distinct normalisation thresholds are derived for each stimulus, depending on its typical raw output range. Certain stimuli (e.g. stuck) are also inverted (i.e. 0↔1 and the direction reversed). Normalised stimuli S_{norm} are unit interval variables, so inverting a stimulus is a simple subtraction:

$$S_{norm} \leftarrow 1 - S_{norm} \tag{8.2}$$

Next, the normalised stimulus values S_{norm} are filtered to produce damped values S_{damp} using a simple weighted average function:

$$S_{damp} \leftarrow \alpha S_{norm} + (1 - \alpha) S_{damp} \tag{8.3}$$

The damping factor α controls a stimulus's temporal characteristics. Separate damping factors are employed for the growth (α_1) and decay (α_2) of each stimulus. For example, some stimuli require rapid growth but slow decay (represented by high α_1 and low α_2), while the reverse is true for other stimuli.

8.1.1 Danger

For a mobile robot, the primary danger is that it may collide with obstacles, resulting in damage to itself and/or its environment. Hence, in our implementation, the danger stimulus is a function of obstacle proximities. It utilises the obstacle distance vector field produced by the directional controller. The distance associated with each vector is taken into consideration, as well as its direction relative to the front of the robot. Obstacles close to the front of the robot are considered more dangerous than those behind it, so the distance d_o of each vector is increased by a function of the angle between it and the frontal direction θ_F:

$$D_o(\theta) = \frac{d_o(\theta)}{d_{D(\max)}} + \kappa_\theta \frac{|\theta_F - \theta|}{\pi} \tag{8.4}$$

This results in a modified vector field $\mathbf{D_o}$ that is biased towards directions near the front of the robot (Figure 8.1). Increasing the direction factor κ_θ causes $\mathbf{D_o}$ to be more heavily biased towards frontal directions (Figure 8.2). The threshold distance $d_{D(\max)}$

alters the level of influence that obstacle distance has over the vector field. Lower values decrease the perceived danger of nearby obstacles (Figure 8.3).

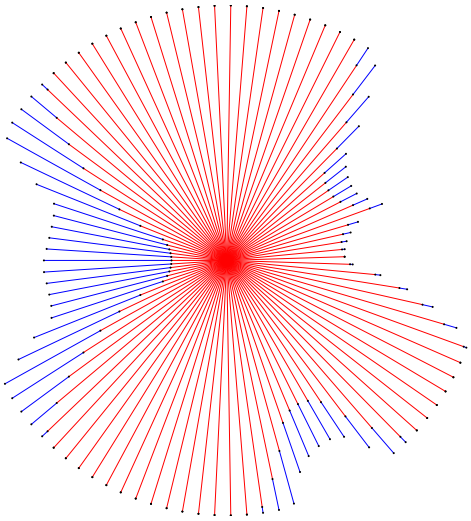

Figure 8.1: Modified vector field (blue) with κ_θ = 0.5 and $d_{D(\max)}$ = 3 m. The normalised, but otherwise unmodified obstacle distance vector field (red) is superimposed on top of it. The robot's heading is directly east.

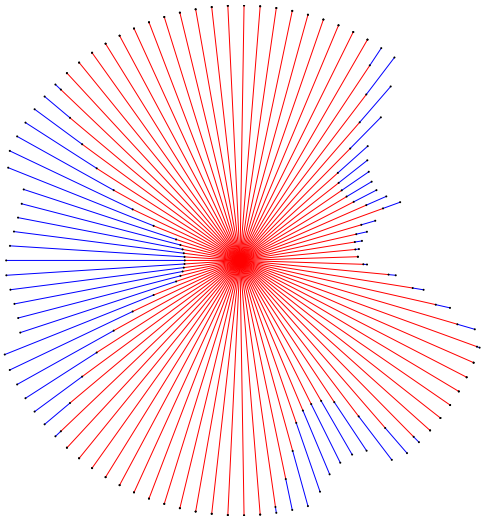

Figure 8.2: Modified vector field with κ_θ = 0.7 and $d_{D(\max)}$ = 3 m.

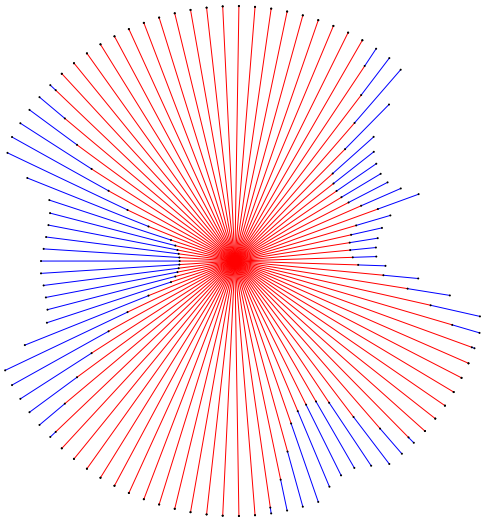

Figure 8.3: Modified vector field with κ_θ = 0.5 and $d_{D(\max)}$ = 2.5 m.

The raw value of the danger stimulus S_{danger} is a function of the minimum of the vector field $\mathbf{D_o}$:

$$S_{danger} = 1 - \min(\mathbf{D_o}) \tag{8.5}$$

The normalisation thresholds of this stimulus are simply 0 and 1, as it already typically resides in the intended interval. Growth is less severely damped than decay,

to allow a rapid response to perceived threats, while preventing the stimulus from oscillating rapidly between high and low values.

8.1.2 Stuck

The stuck stimulus detects obstructed states that are preventing the robot from achieving its goals. In the context of navigation, this is represented by a lack of goal-directed movement. The stimulus can be triggered if the robot literally comes to a standstill, but it can also result from repetitive motion that fails to make sufficient progress in any direction (either towards the goal point or otherwise). Cyclical or otherwise repetitive motion indicates a potential infinite loop that the robot should attempt to escape.

A raw stimulus variable S_{stuck} is obtained that represents the robot's overall speed over time interval $t = [t_0, t_n]$:

$$S_{stuck} = \frac{\left\| \sum_{i=0}^{n-1} (t_{i+1} - t_i)\mathbf{v}(t_i) \right\|}{t_n} \tag{8.6}$$

The magnitude of each linear velocity vector $\mathbf{v}(t_i)$ is the robot's speed at time t_i, while the direction is its heading at time t_i. Hence, the upper term is an approximation of the robot's total displacement during time t_n. The value of t_n influences the amount of time the robot must linger in a given location before it detects an obstructed state. If t_n is low, the stuck stimulus is more transient, whereas higher values slow down its response.

Mobile robots rarely move in straight lines for significant periods of time, so the raw S_{stuck} value represents a speed that is typically lower than the robot's average speed. It is normalised to cover the interval [0, 1], and inverted using Equation 8.2 so that higher speeds correspond to lower S_{stuck} intensities. Normalisation and inversion result in a variable that is usually 0, but which increases towards 1 when the robot's overall velocity (the raw S_{stuck} value) drops below a threshold.

A very low growth factor is applied to this stimulus, while its decay is undamped. This reduces the rate of false-positives when detecting obstructed states. The robot must consistently fail to escape an obstruction before it determines that a behavioural change is required.

8.1.3 Pain

Pain can be viewed as a mechanism that discourages an adaptive system from repeating or continuing actions that cause it physical harm. For mobile robots, the most obvious source of pain is a collision. Since a robot typically does not have nerves, it cannot easily detect physical damage (unless the damage can be inferred from measurably diminished capabilities). Instead, our implementation of pain is a simple abstraction that increases when a collision occurs, and decreases over time. The raw S_{pain} stimulus is proportional to the robot's linear velocity v at the time of collision:

$$S_{pain} = \begin{cases} v & \text{if collision} \\ 0 & \text{otherwise} \end{cases} \tag{8.7}$$

Higher levels of pain result from high-speed collisions, because the risk of physical damage is increased. It does not account for the relative speeds or masses of objects with which the robot collides, however. This simple abstraction could be extended by incorporating an actual measure of the forces involved during a collision, but it would require additional hardware (e.g. accelerometers) and software.

The stimulus is normalised so that a relatively high-speed collision yields an S_{pain} intensity of 1. It is filtered such that increases are undamped, but decreases are highly damped. Thus, pain instantly increases when a collision occurs, but it decays slowly in the absence of collisions.

8.1.4 Achievement

Achievement is a measure of progress towards the robot's goal. Navigation progress is expressed in the form of a speed that considers only the robot's overall movement towards or away from the goal point. The raw stimulus $S_{achieve}$ obtains this speed from the initial distance to the goal point $d_g(t_0)$, the current distance $d_g(t_c)$, the time t_0 when the navigation instruction was received, and the current time t_c:

$$S_{achieve} = \frac{d_g(t_0) - d_g(t_c)}{t_c - t_0} \tag{8.8}$$

Motion tangential to the direction of the goal point does not contribute to the speed, because it does not affect the current distance to goal $d_g(t_c)$. Normalisation of $S_{achieve}$ causes appropriate velocities to span the interval [0, 1]. Growth and decay of this

stimulus are damped to limit its transient behaviour following a transition from one navigation instruction to the next.

8.1.5 Density

The density stimulus is an estimate of the obstacle density of explored space. From a navigation perspective, a high obstacle density generally indicates a more challenging environment. The raw stimulus $S_{density}$ is a weighted average of occupancy probabilities $p_o(x_i)$ of nodes x_1-x_n stored in the occupancy grid map, where the weights are exploration probabilities $p_e(x_i)$ obtained from the exploration grid map:

$$S_{density} = \frac{\sum_{i=1}^{n} p_e(x_i) p_o(x_i)}{\sum_{i=1}^{n} p_e(x_i)} \tag{8.9}$$

Different environments tend to produce relatively small deviations in $S_{density}$. Hence, even though it is already a unit interval variable, $S_{density}$ requires normalisation to cover the entire interval [0, 1]. The normalisation thresholds are tuned to account for the $S_{density}$ ranges produced by the robot's intended environments. Density is undamped, because its global nature renders it comparatively unsusceptible to oscillatory behaviour.

8.1.6 Learning

This stimulus is an estimate of the rate of knowledge acquisition. In our navigation architecture, this does not relate to formal learning methods such as reinforcement learning. Rather, it involves dynamically updating the robot's deliberative maps to incorporate new environmental and representation data. This knowledge is represented quantitatively in the form of the exploration grid map.

Raw stimulus variable $S_{learning}$ is the mean of all exploration probabilities p_e in a given map divided by the time t_e spent exploring the map:

$$S_{learning} = \frac{\overline{p_e}}{t_e} \tag{8.10}$$

This is typically a very small number even under ideal conditions (travelling rapidly through a previously-unexplored environment), so to cover the appropriate interval, $S_{learning}$ is normalised using a very low upper threshold. Since it is a function of data

averaged over an entire map, the stimulus is not prone to rapid fluctuations, so it does not require any damping.

8.1.7 Mismatch

The mismatch stimulus indicates a perceived discrepancy between the robot's stored knowledge about its local environment and its current sensor data. This can occur if the environment has not yet been explored, or if the occupancy grid map is sufficiently inaccurate due to environmental dynamics or localisation failures.

A raw stimulus S_{mism} is obtained by comparing instantaneous occupancy status $s_o(x_i)$ of nodes x_1-x_n (where n denotes the number of nodes updated from sensor data) to their stored occupancy probabilities $p_o(x_i)$:

$$S_{mism} = \frac{\sum_{i=1}^{n} w(x_i)\left|s_o(x_i) - p_o(x_i)\right|}{\sum_{i=1}^{n} w(x_i)} \tag{8.11}$$

Nodes whose occupancy status $s_o(x_i)$ is higher than their occupancy probabilities $p_o(x_i)$ are weighted higher than others, to reduce the influence of free space nodes. The weight $w(x_i)$ is set to one of two unit interval constants W_{m1} and W_{m2} (where $W_{m1} < W_{m2}$) depending on whether or not $s_o(x_i)$ is greater than $p_o(x_i)$:

$$w(x_i) = \begin{cases} W_{m1} & \text{if } s_o(x_i) < p_o(x_i) \\ W_{m2} & \text{otherwise} \end{cases} \tag{8.12}$$

Figure 8.4 shows a portion of an occupancy status map obtained from instantaneous sensor data, while the corresponding occupancy grid map is represented in Figure 8.5. The resulting magnitude of the difference between them for each node is shown in Figure 8.6. Sensor noise and filtering delays typically result in a significant amount of disagreement between the two representations even if the environment has been fully mapped. However, it tends to be greater when the robot is exploring a previously-unmapped area.

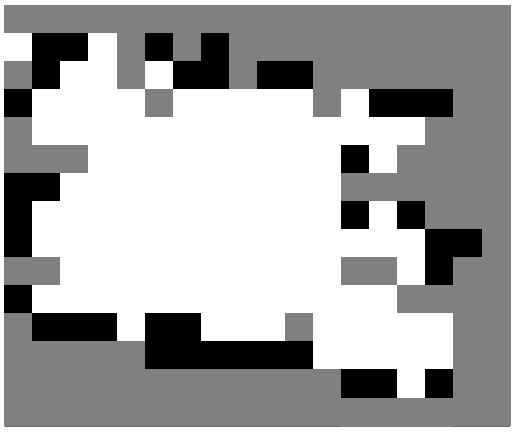

Figure 8.4: Example of occupancy status.

Figure 8.5: Example of occupancy probabilities.

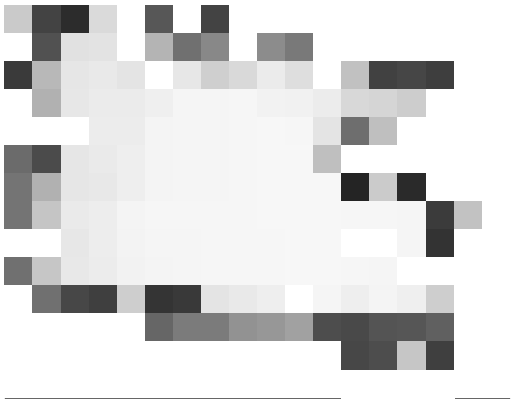

Figure 8.6: Magnitudes of the differences between occupancy status and probabilities.

The stimulus is heavily normalised because its raw value typically occupies a relatively small interval. Increases to S_{mism} are undamped, while decreases are damped, as the stimulus is otherwise susceptible to oscillatory behaviour.

8.1.8 Cost

Cost is a stimulus that increases if the paths planned by the robot are of low quality. Low quality paths are paths that travel through undesirable nodes. Such paths are only utilised if there are no desirable options available. This could be caused by environmental features (e.g. if a dynamic obstacle blocks a doorway for a sufficient duration to increase the occupancy probabilities of nodes in the doorway). It may also result from certain internal configurations (e.g. highly weighted danger, exploration and/or emotion maps producing very high node costs), or a combination of the two.

Low quality paths are often unviable (e.g. passing through walls), so the robot should adjust its planning weights to reduce the contributions of non-occupancy grid maps.

The raw stimulus S_{cost} is the highest cost value from the set of nodes m_p contained within the planned path:

$$S_{cost} = \max(m_p) \tag{8.13}$$

Each member of m_p is the same node cost that is utilised to plan the path, so it is fused from all grid maps and filtered to take into account the costs of surrounding nodes.

The stimulus is normalised such that the lower threshold represents a maximum acceptable cost, and the upper threshold represents the cost of a completely unviable path. Hence, the normalised stimulus ordinarily has an intensity of 0, but it increases upon detection of a low-quality path. Decreases are damped to reduce any rapid fluctuations that may occur if the replanning rate is very high.

8.1.9 Error

Although its name implies a general error detection mechanism, this stimulus is presently only employed to detect internal conflicts between different sensing modalities. Its scope is further constrained by the fact that only two types of exteroceptive sensors are currently modelled in our simulation: infrared distance-measuring sensors and collision sensors. Thus, the stimulus simply responds to collisions with obstacles that are not detected by the infrared sensors. This situation typically arises due to sensor failures or limitations (such as an inability to detect objects outside of a certain plane of detection, or objects constructed from transparent materials).

If a collision occurs, the raw stimulus S_{error} is equal to the distance d_c between the position of the collision sensor triggered by the collision and the closest obstacle position measured by the infrared sensors:

$$S_{error} = \begin{cases} d_c & \text{if collision} \\ 0 & \text{otherwise} \end{cases} \tag{8.14}$$

This raw value is normalised using a lower threshold that represents the minimum distance required for the sensors to ‘definitely agree’, and an upper threshold representing a distance above which the sensors ‘definitely disagree’. Increases to S_{error} are undamped to facilitate a rapid response to perceived sensor inconsistencies,

while decreases are highly damped so that the robot continues to act to prevent further collisions with undetectable obstacles.

8.1.10 Stimulus Parameters

Specific parameter values for each stimulus are presented in Table 8.1. These parameter values are typically obtained by manual observation and adjustment to produce the intended response, rather than by quantitative analysis of their effects on the robot's performance. They are small components of a secondary system which itself exerts a relatively subtle influence over the robot's behaviour. Small incremental changes are not typically measureable over noise (but large changes are measureable, as shown by the results in Chapters 9 and 10). Thus, the parameter values should be regarded as adequate, but not necessarily optimal. Individual justifications for these parameters are provided in the preceding sections. The damping factors assume a control period of 0.1 s, and would need to be adjusted accordingly for different sampling intervals.

TABLE 8.1: STIMULUS PARAMETERS

Stimulus	**Normalisation Thresholds**		**Damping Factors**		**Other Parameters**
	S_{min}	S_{max}	α_1	α_2	
Danger	0	1	0.1	0.05	$\kappa_\theta = 0.5$ $d_{D(max)} = 1.5$ m
Stuck	0.1 m/s	0.2 m/s	0.005	1	$t_n = 30$ s
Pain	0 m/s	0.5 m/s	1	0.01	
Achievement	0.1 m/s	0.4 m/s	0.1	0.01	
Density	0.15	0.25	1	1	
Learning	0	0.002	1	1	
Mismatch	0.3	0.35	1	0.05	$W_{m1} = 0.5$ $W_{m2} = 1$
Cost	0.5	0.7	1	0.01	
Error	0.3 m	0.5 m	1	0.01	

8.2 Drives

Drives modulate the robot's planning and control parameters in response to emotions and moods. Each drive is responsible for a subset of parameters that cause similar behavioural changes. The values of parameters governed by a given drive are assigned

positions on a spectrum between two opposing 'modes of behaviour' (e.g. exploration/exploitation). These positions are determined by a unit interval variable D representing the intensity of the parameters' governing drive.

The conversion from a drive intensity D to a parameter value P is typically a simple linear mapping from an input variable with interval [0, 1] to an output variable with interval $[P_1, P_2]$ or $[P_2, P_1]$ (if $P_2 < P_1$). However, linear conversion is not ideal for certain parameters. A modified drive intensity D' is utilised to enable curved input-output relationships in addition to linear ones. The shape of the function is defined by the modulation exponent X:

$$D' = \begin{cases} 1-(1-D)^{-X} & \text{if } X < -1 \\ D^{X} & \text{if } X > 1 \\ D & \text{otherwise} \end{cases} \tag{8.15}$$

As the magnitude of X increases above 1, the relationship becomes increasingly curved, as shown in Figure 8.7.

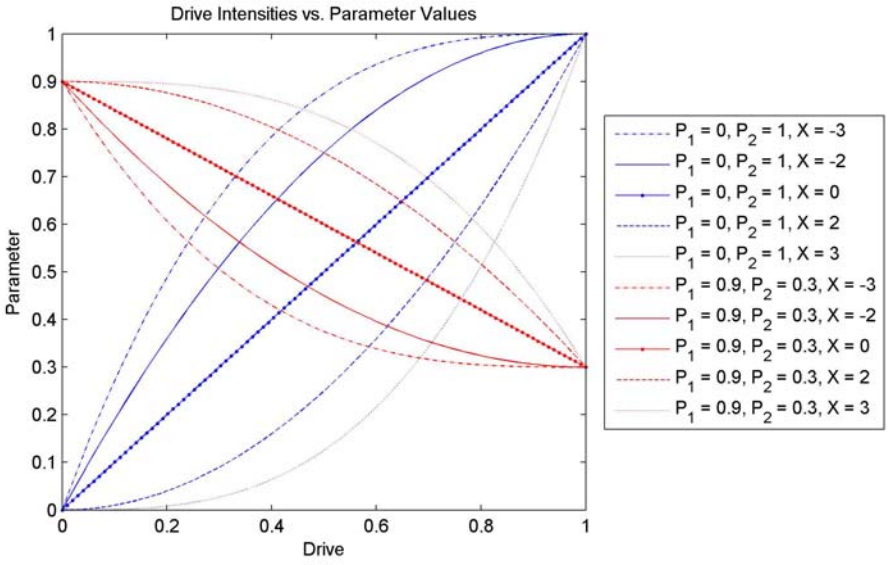

Figure 8.7: Examples of drive/parameter relationships with different P_1, P_2 and X values.

The modified value D' is applied to the conversion from drive to parameter value P:

$$P = (1-D')P_1 + D'P_2 \tag{8.16}$$

In Chapter 7, individual parameters associated with each drive were analysed, and their quantitative effects on performance were identified. Appropriate modulation

ranges for each parameter derived from these experimental results are shown in Table 8.2.

These can be divided into survival and strategic drives. Survival drives (safety, speed and efficiency) must constantly make tradeoffs between the robot's competing needs. Thus, they are rarely set to 0, and their modulations are naturally centred on an equilibrium position of 0.5. Conversely, strategic drives (exploration, action and introspection) are more situational in their applications, and they may be inactive for significant durations. Hence, the strategic drives are considered at equilibrium when set to 0, and they are increased above that equilibrium position only when necessary.

TABLE 8.2: DRIVE PARAMETERS

Drive	**Parameter**	**Modulation range**		**Modulation exponent**
		P_1 (D = 0)	P_2 (D = 1)	
Safety	W_{v1}	0.9	0.99	0
	W_{v4}	0.6	0.4	0
	r_o	0.25 m	0.35 m	0
	t_2	0.9 s	1.1 s	0
Speed	v_L	0.4 m/s	1.0 m/s	0
Efficiency	n_o	7	1	0
	n_θ	30	10	0
	t_r	0.1 s	2.0 s	0
Exploration	W_{p3}	0	0.4	0
	ε_2	0.01	0.02	0
Action	$W_{\theta 3}$	0.75	0.25	10
	$W_{\theta 5}$	0	0.5	-1.5
	$W_{\theta 6}$	0	0.75	-1.5
	d_L	1.05 m	2.45 m	10
Introspection	W_{v2}	0	0.9	0
	$W_{\theta 2}$	0	0.99	0
	W_{p2}	0	1	0

8.2.1 Safety

Safety drive D_{safety} influences trade-offs between ensuring that the robot's motion is collision-free and overcoming environmental challenges to converge on the goal point. Velocity controller avoidance weight W_{v1} is decreased proportionately with D_{safety}, to allow the robot more freedom in choosing velocities in borderline situations (where collisions are unlikely, but possible). Experimental results in Section 6.3 show that a value of 0.9 does not significantly increase the collision count compared to a

value of 0.99, but it does improve performance slightly. Speedup weight W_{v4} increases as D_{safety} diminishes, encouraging the robot to select higher linear velocities. This can improve the robot's ability to traverse narrow doorways, but it comes at a slight cost to stability and safety. Values in the range 0.4 – 0.6 typically result in viable, collision-free motion. If W_{v4} is outside of this range, the robot either becomes so easily obstructed that it rarely converges on the goal point, or its safety is severely compromised.

Obstacle radius r_o is decreased proportionately with D_{safety}, enlarging the perceived size of openings such as doorways, enabling the robot to traverse them more readily. Although the minimum radius of 0.25 m is less than that of the robot, the results in Section 7.1.1 show that this value does not result in constant collisions. The robot favours directions and velocities that are further from obstacles regardless of their perceived size. Nonetheless, r_o is in principle a hard limit that might otherwise prevent certain collisions. Curvature time t_2 has a narrow viable range of 0.9 – 1.1 s, as shown in Section 7.1.3, but modulations are enabled within this range. Increasing D_{safety}, yields a proportionate increase to t_2, causing the robot to slow down or turn away from obstacles earlier upon encountering them. This may prevent some collisions, but it can restrict the velocity controller's options to the point where it becomes obstructed.

8.2.2 Speed

Speed drive D_{speed} controls two important kinematic/dynamic constraints applied to the velocity controller. First is the linear velocity limit v_L, which represents the maximum linear velocity allowed by the robot, disregarding its dynamic constraints. It is increased proportionately with D_{speed} over the range 0.4 – 1.0 m/s. In theory, higher v_L values yield faster goal convergence. However, in practice, performance improvements arising from higher speeds suffer from significant diminishing returns, depending on the robot's environment, as shown in Section 7.2.1. Cluttered indoor environments rarely allow a robot to achieve its maximum speed. Furthermore, navigational problems due to higher speeds (e.g. missed turns into doorways) tend to outweigh any short-term improvements. Sparsely occupied outdoor environments generally allow the robot to benefit more from higher speeds. Regardless, the robot

suffers from an increased risk of collisions at higher speeds, and the potential damage sustained during a collision also increases.

8.2.3 Efficiency

Efficiency drive D_{effic} modulates parameters controlling the level of computational effort applied to navigation. Obstacle buffer size n_o decreases as D_{effic} increases, reducing the amount of obstacle data processed by the reactive control layers. Results presented in Section 7.3.3 show that this produces faster execution speeds at a potential cost to safety. A range of 1 – 7 is selected because the collision rate shows an improvement over this range, but it becomes largely constant when $n_o > 7$. Similarly, vector field size n_θ is inversely and linearly related to D_{effic}. As shown in Section 7.3.2, decreasing n_θ tends to negatively impact goal convergence while linearly improving computational efficiency. It is modulated over the range 10 – 30 because values lower than 10 yield unacceptable performance characteristics, while values higher than 30 decrease efficiency while providing no discernable improvements to goal convergence.

Increases to the replan period t_r are proportional to rises in D_{effic}, as less frequent replans in theory yield improved efficiency. In practice, this only holds true for $t_r < 2$ s, so t_r is modulated over the range 0.1 – 2 s (where the default control period is 0.1 s). A very small improvement to convergence speed is apparent over this range (Section 7.3.4), so the costs and benefits of changing t_r are relatively minor.

8.2.4 Exploration

Exploration drive $D_{explore}$ represents the dichotomy between exploration (obtaining new world knowledge to improve long-term prospects) and exploitation (utilising existing world knowledge to maximise short-term rewards). Two parameters modulated by this drive are the planning exploration weight W_{p3} and the occupancy growth rate ε_2. Exploration weight W_{p3} increases linearly with $D_{explore}$ over the range 0 – 0.4. Experimental results in Section 7.4.1 show that W_{p3} growth within this range yields a consistent improvement to exploration coverage, while also decreasing convergence speed (due to less direct paths to the goal being planned). Values greater than 0.5 tend to degrade the quality of planned paths, sometimes preventing the robot

from reaching the goal. However W_{p3} is limited to values lower than 0.4 so that once the exploration map is fused with emotion maps (described in Section 8.3.2 below), their combined maximum value is around 0.5.

Occupancy growth rate ε_2 is modulated over the range 0.01 – 0.02 by $D_{explore}$. At is peak, the growth rate is double the decay rate ε_1. Performance improvements resulting from constant values within this range are unclear according to the results in Section 7.4.2 (nor are any significant decreases apparent). Nevertheless, in theory, the robot may benefit slightly from temporary increases to ε_2 while it is constructing its map (and experimental results show that it is unlikely to adversely affect performance). Hence, it is proportionately linked to $D_{explore}$.

8.2.5 Action

Action drive D_{action} tilts the balance between reactive control and deliberative control by increasing either the directional controller's path following weight $W_{\theta 3}$ or its goal seeking, angular inertia and wander weights $W_{\theta 4}$, $W_{\theta 5}$, and $W_{\theta 6}$. Experimental results in Sections 6.2 and 6.4 show that the robot's performance is improved when either $W_{\theta 3}$ is set to 0.75, or when $W_{\theta 5}$, and $W_{\theta 6}$ are set to 0.5 and 0.75, respectively. If D_{action} were to vary the weights linearly, both sets of weights would be significantly lower than their optimal values near the centre of the transition, potentially resulting in decreased performance. Instead, the 'primary' weight(s) are maintained close to their peak values until 'secondary' weight(s) have increased to the levels required to assume control over the robot's behaviour and become 'primary'. This behaviour is achieved using modulation exponents of -1.5 and 10 (depending on the direction of modulation).

The look-ahead distance d_L is also assigned a non-zero modulation exponent (E = 10) so that it increases very slowly until the robot is heavily biased towards reactive control, as performance is otherwise unnecessarily reduced.

8.2.6 Introspection

Introspection drive D_{intro} controls the balance between reactive and deliberative world representations utilised for obstacle avoidance. Parameters affected include the velocity and directional controllers' sensor-based (W_{v1} and $W_{\theta 1}$) and map-based (W_{v2}

and $W_{\theta 2}$) avoidance functions. As D_{intro} grows, W_{v2} and $W_{\theta 2}$ are increased linearly over the ranges 0 – 0.9 and 0 – 0.99, respectively. The different ranges are selected for practical reasons; if $W_{v2} \approx W_{\theta 2}$, the robot tends to become obstructed by undetectable obstacles when it gets too close to them. This is caused by a conflict between the directional and velocity controllers previously described in Section 7.6.1 (in relation to the equivalent sensor-based avoidance functions).

The final parameter to be modulated by D_{intro} is the danger weight W_{p2}, which controls the contribution of the danger grid map to path planning. When moderately activated, this can prevent collisions with unseen obstacles, as shown in Section 7.6.2. However, it could potentially cause doorways to become blocked (from the perspective of the path planner) if it were constantly enabled. Hence, it grows linearly within the range 0 – 1 in response to increases of D_{intro}.

8.3 Emotions

A set of discrete emotions is implemented to control the robot's drives appropriately in response to its various stimuli. While they may bear only a superficial resemblance to biological emotions with the same names, the emotional labels are useful for encapsulating certain types of adaptive behaviour.

The emotions modelled in our system can be divided into two interacting components. First are global emotions, which are short-term intensities elicited by certain combinations of stimuli. Second are mapped emotions, which are associated with specific locations in the environment. Separate emotion intensity values are stored for each map node, enabling location-specific parameter modulations and biases to deliberative path planning. The robot's drives are controlled by weighted combinations of both global and mapped emotions.

8.3.1 Global Emotions

Each global emotion E_G is a function of multiple affective stimuli. The mechanism employed to combine stimuli is a dynamic weighted sum that can perform different 'logical' operations on the inputs. This function resembles a simplified Sugeno fuzzy inference system (Sugeno, 1985) containing a single linear membership function and an independent if-then rule for each input. The membership function for an input I_i is

represented by minimum and maximum weights $W_{i(\min)}$ and $W_{i(\max)}$, which determine the weight W_i if I_i is 'disabled' and 'enabled', respectively. Four different fuzzy operations are represented by logic tag L_i:

- $L_i = 0$: If I_i is high, output E_G is high.
- $L_i = 1$: If I_i is high, E_G is low.
- $L_i = 2$: If I_i is low, E_G is high.
- $L_i = 3$: If I_i is low, E_G is low.

The input I_i can be inverted such that a higher value yields a lower output E_G. This yields a factor F_i that is utilised in the final calculation:

$$F_i = \begin{cases} (1 - I_i) & \text{if } (L_i = 1) \vee (L_i = 2) \\ I_i & \text{otherwise} \end{cases} \tag{8.17}$$

A multiplier M_i is also calculated from input I_i and exponent X_i:

$$M_i = \begin{cases} (1 - I_i)^{X_i} & \text{if } (L_i = 2) \vee (L_i = 3) \\ I_i{}^{X_i} & \text{otherwise} \end{cases} \tag{8.18}$$

The weight W_i is a function of thresholds $W_{i(\min)}$ and $W_{i(\max)}$, and multiplier M_i:

$$W_i = (1 - M_i) W_{i(\min)} + M_i W_{i(\max)} \tag{8.19}$$

Once weights W_i and factors F_i have been calculated for all indices i, the output E_G is obtained using a weighted average calculation:

$$E_G = \sum_{j=1}^{n} \left(\frac{W_j F_j}{\sum_{i=1}^{n} W_i} \right) \tag{8.20}$$

Figure 8.8 shows an example of the output generated by a two-input dynamic weighted sum function. The first input is considered dominant, while the second input can increase the minimum value of the output. However, second input values near the middle of its [0, 1] interval (e.g. 0.5) also decrease the output's maximum value, which is not intended (ideally, it should remain constant). This disturbance can be reduced by increasing the value of X_i, but it also changes the shape of the function when the first input is lower (Figure 8.9).

In order to eliminate the high-end disturbance without adversely affecting the low-end, the contribution of the second input is reduced when the first is highly weighted.

A dominant input $i = D$ is indicated by a $W_{i(\min)}$ value of 1. If none of the inputs satisfy this criterion, none are considered dominant and this step is omitted. The multipliers M_i of all other inputs are modified by the following equation:

$$M_i \leftarrow \begin{cases} (1 - F_D) M_i & \text{if } (L_i = 1) \vee (L_i = 3) \\ F_D M_i & \text{otherwise} \end{cases} \tag{8.21}$$

The result is a form such as that shown in Figure 8.10, which has the desired output characteristics over the full dominant input range.

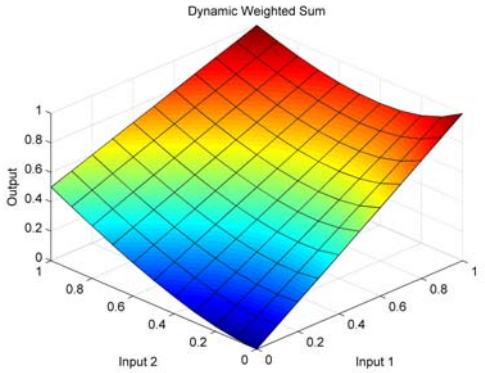

Figure 8.8: Example unmodified dynamic weighted sum function. W_{min} = (1, 0), W_{max} = (1, 1), X = (0, 0), L = (0, 0).

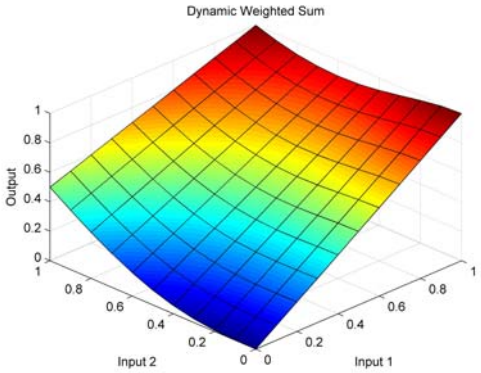

Figure 8.9: Example unmodified dynamic weighted sum function. W_{min} = (1, 0), W_{max} = (1, 1), X = (0, 2), L = (0, 0).

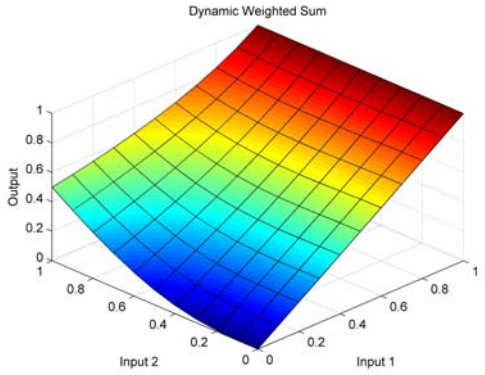

Figure 8.10: Example modified dynamic weighted sum function. W_{min} = (1, 0), W_{max} = (1, 1), X = (0, 2), L = (0, 0).

Specific weights and logic tags for each stimulus-to-emotion conversion are shown in Table 8.3. The linkages and parameters utilised in our implementation are not the only valid options. Human-like emotions may require more complex context-dependent interconnections, while robot performance equivalent to ours may be achieved by a simpler configuration (e.g. one stimulus per emotion). Our configuration can be

regarded as a compromise between the requirement for ostensibly nondeterministic behaviour, and the need for demonstrable performance improvements.

TABLE 8.3: GLOBAL EMOTION PARAMETERS

Emotion	**Eliciting Stimulus**	**Membership function**		**Exponent**	**Logic tag**
		W_{min}	W_{max}		
Anger	Stuck	1	1	0	0
	Achievement	0	0.5	0	1
Fear	Danger	1	1	0	0
	Pain	0	2	0	0
Happiness	Achievement	1	1	0	0
	Density	0	1	0	1
Sadness	Pain	0.2	1	2	0
	Error	0.2	1	2	0
Curiosity	Learning	1	1	0	0
	Cost	0	4	2	1
Surprise	Mismatch	1	1	0	0
Confusion	Error	1	1	0	0

Anger (Figure 8.11) arises if the stuck stimulus is high (indicating a current obstructed state), but it is reduced if achievement is also high. Thus, anger is generally lower if the robot has recently made rapid progress. It cannot reach its maximum intensity unless it continues to fail to converge on the goal. This helps prevent the robot from reducing its safety margins further than are necessary.

Fear (Figure 8.12) is a function of the danger and pain stimuli. Danger indicates that a collision is likely to occur, while pain signifies a recent collision. Although biological pain is more often connected to anger than fear (Izard, 1993), it is counterproductive in our implementation for a robot to increase an emotional state that disregards safety at a time when safety has already been sufficiently compromised to allow a collision.

Happiness (Figure 8.13) is elicited by a high achievement stimulus, which occurs when overall progress towards the goal point is rapid. This is an indication that navigation through the current environment is unproblematic. Happiness is also inversely dependent on the density stimulus. High occupancy densities signify a potential navigational challenge, lowering happiness. Both stimuli are assigned equal weights, as they are both useful indicators of the level of computational effort the robot should devote to navigation.

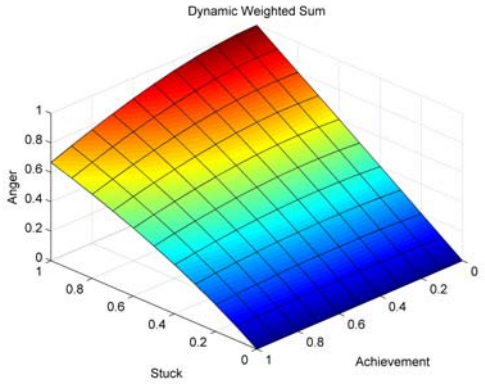

Figure 8.11: Anger elicited by stuck and achievement stimuli.

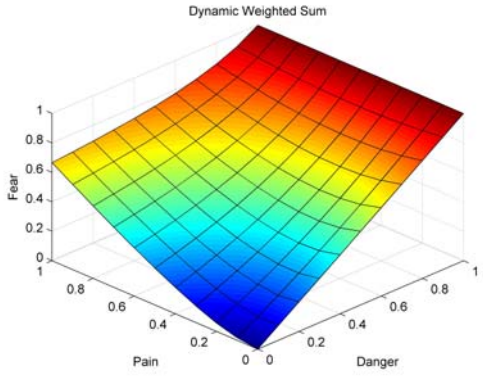

Figure 8.12: Fear elicited by pain and danger stimuli.

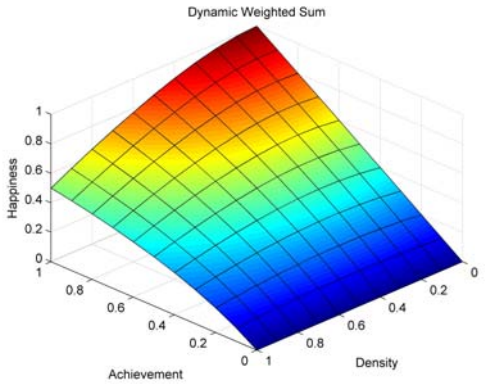

Figure 8.13: Happiness elicited by achievement and density stimuli.

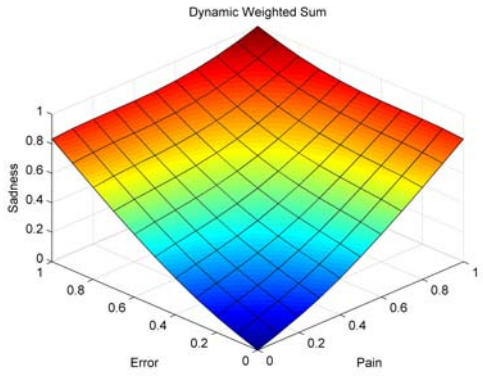

Figure 8.14: Sadness elicited by error and pain stimuli.

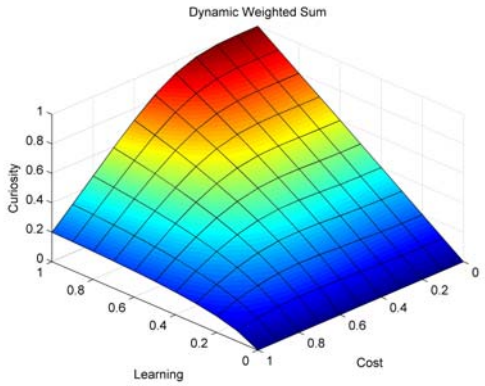

Figure 8.15: Curiosity elicited by learning and cost stimuli.

Sadness (Figure 8.14) is induced by the pain and error stimuli, both of which are caused by collisions in our implementation. While pain is dependent on the robot's speed immediately prior to a collision, error is a measure of how closely a point of collision matches obstacle positions measured by the robot's primary sensors. Thus,

sadness is caused by a failure to avoid a high-speed collision (due to control limitations) and/or a failure to detect an imminent collision (due to sensor limitations).

Curiosity (Figure 8.15) increases due to the learning stimulus, and decreases in response to the cost stimulus. Learning is a global estimate of the exploration rate of a whole environment. Hence, curiosity is the only emotion whose primary stimulus does not change significantly depending on the robot's location. Its main response is to modulate the exploration drive. When this drive is highly activated, the quality of the robot's paths may be degraded. Thus, heavy reductions are applied to curiosity if the cost stimulus determines that the planned path is unviable.

Surprise occurs due to a perceived mismatch between reality and predictions, represented by the mismatch stimulus. In practical terms, it means that the robot's local sensor data do not match its occupancy grid map, so either the environment is not accurately known, or the robot's localisation information is incorrect. Thus, a temporarily reduced emphasis on deliberative planning may be beneficial.

Confusion is dependent on the error stimulus, indicating that the robot's sensors are unreliable for obstacle detection. It therefore becomes necessary to rely more heavily on deliberative internal representations to prevent further collisions.

8.3.2 Mapped Emotions

Mapped emotions are associated with specific grid map nodes. Each mapped emotion grows or decays in nodes x_1-x_n close to the robot's current position. Over time, the intensity values $e_M(x_i)$ of a node x_i tends towards the robot's equivalent global emotion intensity E_G. The rate of growth or decay is a function of a node's Euclidean distance $d(x_i)$ from the robot's position, and a damping factor γ (again assuming a control period of 0.1 s):

$$e_M(x_i) \leftarrow \gamma\left(1-\frac{d(x_i)^2}{{r_M}^2}\right)E_G + (1-\gamma)e_M(x_i) \tag{8.22}$$

Nodes close to the robot's position are more strongly influenced by E_G than nodes further away (Figure 8.16). The radius of influence is represented by r_M. Enlarging r_M increases the influence exerted on more distant nodes, and it also expands the total number of nodes affected. The damping factor γ is substituted with a distinct growth factor γ_1 or decay factor γ_2, whose selection depends upon whether or not $e_M(x_i) < E_G$.

These factors control the overall increase and decrease of $e_M(x_i)$ values, regardless of their positions relative to the robot.

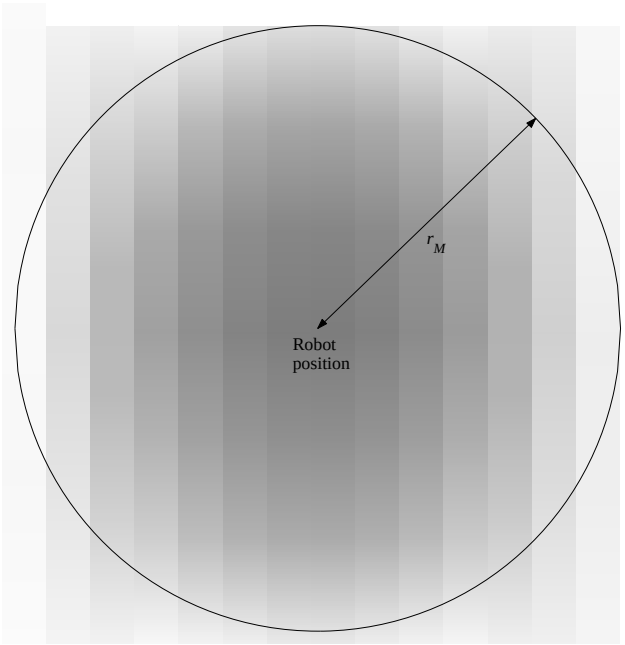

Figure 8.16: Updating an emotion map from a global emotion intensity.

Emotion maps can be utilised as path planning biases in an identical manner to the exploration and danger grid maps. Positive and negative emotions are analogous to attractive and repulsive forces applied to specific map regions. Negative emotions increase the cost of nodes where they are strongly elicited. They are fused together by Equations 5.3 and 5.4. Positive emotion maps are also fused with each other and the resulting intensities are inverted such that higher intensities reduce node costs. Next, the negative and positive maps are fused to form a single emotion map. During this fusion, the positive emotion map is weighted by the robot's current positive mood, while the negative emotion map is weighted by its negative mood (these moods are described in the Section 8.4). Both mood contributions are also modulated by the inverted cost stimulus, limiting emotional influences to planning when they adversely influence the quality of planned paths. Finally, the fused emotion map is multiplied by the exploration map, preventing negative biases (due to low positive emotions) from being applied to nodes that the robot has not yet explored. The fused map's overall contribution to path planning is governed by the weight W_{p4}.

Local emotion intensities E_M are obtained from the emotion maps using a weighted average scheme. The weight applied to an individual node x_i is dependent on its proximity to the robot's position:

$$E_M = \frac{\sum_{i=1}^{n}\left(1-\frac{d(x_i)^2}{r_M{}^2}\right)e_M(x_i)}{\sum_{i=1}^{n}\left(1-\frac{d(x_i)^2}{r_M{}^2}\right)} \tag{8.23}$$

This is essentially a reversal of the emotion mapping process represented in Equation 8.19. Similarly, nodes close to the robot exert a stronger influence over E_M than more distant nodes, and the range of influence is controlled by r_M.

Although E_M is a function of E_G, the two emotion intensities can have very different values, particularly if small damping factors γ_1 and γ_2, or a large radius r_M are employed. The global intensity E_G is dependent only on currently perceived stimuli, whereas E_M is a function of previous emotion intensities elicited in the robot near its current position. These are combined by a simple weighted average to form an overall intensity E that interacts with the robot's drives:

$$E \leftarrow \varphi E_G + (1-\varphi)E_M \tag{8.24}$$

Increasing the global/map weight φ shifts the balance towards E_G and away from E_M. This generally improves the robot's responsiveness to a given stimulus, but it decreases its ability to adapt to different environments over time.

Table 8.4 shows the individual parameters assigned to each emotion. By default, the radius r_M is set to 1 m for all emotions. Lower values may negate the benefits of some mapped emotions (e.g. the robot may not attempt to turn away from an unseen obstacle until it arrives at the exact same position where it previously collided with the obstacle). Higher values can also have undesirable consequences (e.g. the robot might respond to an event that occurred on the opposite side of a wall from its current position).

These parameters are chosen based on the individual requirements of each emotion. Some emotions (e.g. anger) are greatly dependent on mapped emotions to perform optimally and thus are assigned low φ values (and in the case of confusion, a value of zero). These emotions generally also have lower γ_2 values to preserve mapped intensities. Conversely, others (e.g. surprise) are more reliant on rapid responses to

immediate stimuli than on mapped information, and thus are assigned higher γ_2 and φ values.

TABLE 8.4: MAPPED EMOTION PARAMETERS

Emotion	**Damping Factors**		**Global/map weight**
	γ_1	γ_2	
Anger	0.1	0.001	0.1
Fear	0.1	0.01	0.5
Happiness	0.1	0.01	0.5
Sadness	0.1	0.001	0.1
Curiosity	0.1	0.1	0.5
Surprise	0.1	0.1	0.5
Confusion	0.1	0.0001	0

8.3.3 Emotional Responses

The combined emotion intensities E are utilised to control the affective drives. Emotions modulate drives in the same manner as stimuli elicit global emotions. The same dynamic weighted sum function is employed (Section 8.3.1). Table 8.5 shows the function's parameters assigned to each emotion-drive combination.

TABLE 8.5: EMOTIONAL RESPONSE PARAMETERS

Drive	**Eliciting emotion**	**Membership function**		**Exponent**	**Logic tag**
		W_{min}	W_{max}		
Safety	Anger	1	1	0	1
	Fear	0	0.25	0	0
	Surprise	0	3	3	0
Speed	Anger	0	1	2	1
	Fear	1	1	0	1
Efficiency	Anger	0	0.25	0	0
	Fear	0	0.25	0	1
	Happiness	1	1	0	0
	Sadness	0	3	2	1
Exploration	Happiness	0	0.25	0	1
	Sadness	0	0.25	0	0
	Curiosity	1	1	0	0
	Surprise	0	1	2	0
Action	Surprise	1	1	0	0
	Confusion	0	1	2	1
Introspection	Confusion	1	1	0	0

Examples of the resulting relationship between the efficiency drive and two of its eliciting emotions are given in Figures 8.17 and 8.18. Efficiency is a function of four input dimensions, so its entire state space cannot be represented graphically. Instead, these figures show the outputs resulting from variations of the two most important inputs (happiness and sadness) when the other inputs (anger and fear) are set at values that produce minimum (Figure 8.17) and maximum (Figure 8.18) outputs.

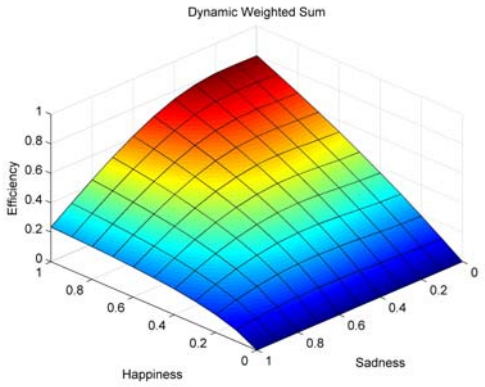

Figure 8.17: Efficiency drive elicited by happiness and sadness, with anger = 0 and fear = 1.

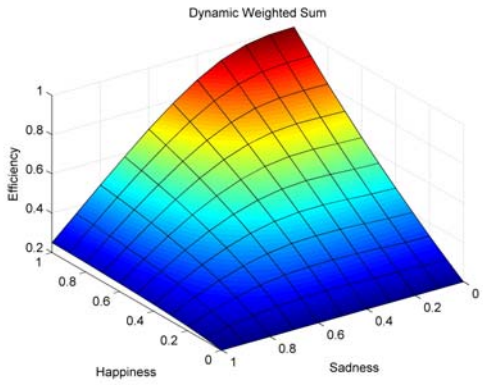

Figure 8.18: Efficiency drive elicited by happiness and sadness, with anger = 1 and fear = 0.

8.4 Moods

Moods are highly damped functions of emotions, so they grow and decay at much slower rates than other affective states. Again, the dynamic weighted sum function is employed to obtain moods from emotions. Table 8.6 shows the damping factors and function parameters for positive and negative moods. The damping factors again assume a control period of 0.1 s. Positive mood is only dependant on happiness, while negative mood is a function of anger, fear and sadness. Figures 8.19 and 8.20 show the undamped negative emotion intensity resulting from two input emotions, with the remaining input set to its minimum and maximum values, respectively.

TABLE 8.6: MOOD PARAMETERS

Mood	Damping factors		Eliciting emotion	Membership function		Expon.	Logic tag
	α_1	α_2		$W_{\min}$	$W_{\max}$		
Positive	0.002	0.001	Happiness	1	1	0	0
Negative	0.002	0.001	Anger	0.2	1	2	0
			Fear	0.2	1	2	0
			Sadness	0.2	1	2	0

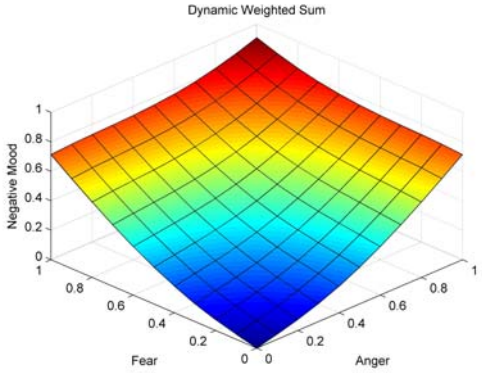

Figure 8.19: Negative mood elicited by anger and fear, with sadness = 0.

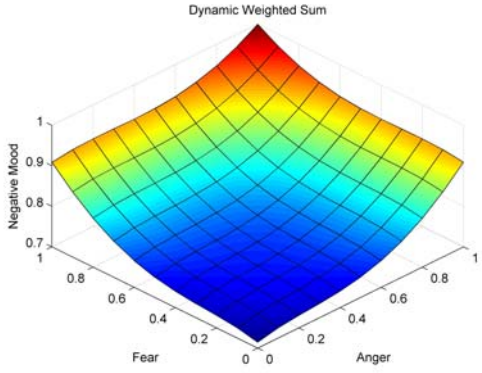

Figure 8.20: Negative mood elicited by anger and fear, with sadness = 1.

The purpose of these moods is to constrain the robot's emotional responses to levels that are appropriate to its environment. They modulate the maximum or minimum values of survival drives (safety, speed and efficiency), affecting the level of influence emotions have over the robot's behaviour. Positive mood M_{pos} varies each drive's upper limit $D_{\max}$ between 0.5 and 1, while negative mood M_{neg} modulates the lower limit $D_{\min}$ between 0.5 and 0:

$$D_{\max} = \frac{1 + M_{pos}}{2} \tag{8.25}$$

$$D_{\min} = \frac{1 - M_{neg}}{2} \tag{8.26}$$

Each emotion-generated survival drive D_E is mapped from the interval [0, 1] to the interval $[D_{\min}, D_{\max}]$, to produce a final drive intensity D:

$$D = D_{\min} + (D_{\max} - D_{\min})D_E \tag{8.27}$$

Strategic drives (exploration, action and introspection) are not constrained by moods. Unlike survival drives, their 'default' position is 0, and restricting their dynamic range is generally counterproductive. Exploration and action are typically most active when

the robot encounters a new environment, at which time its moods and emotions have not yet adapted to the environment. If the environment is fully known, optimal performance is generally achieved if they are set to 0. Introspection can be highly active at any time, but preventing it from spanning the full interval [0, 1] can significantly impede performance and/or safety.

8.5 Summary

The robotic affect model outlined in Chapter 3 has been implemented on the simulated mobile robot. Implementation details for the model's various components (stimuli, drives, emotions and moods) emerged from the methodology and results presented in Chapters 5-7. Stimuli and mapped emotions are functions of specific internal representations described in Chapter 5, such as obstacle distance vectors and grid maps. Drives modulate parameters introduced in Chapter 5, and the parameter categorisations and modulation ranges are derived from results presented in Chapters 6-7. Emotions and moods choose appropriate drive modulations in response to different combinations of stimuli and mapped emotions.

The parameters and interconnections shown in this chapter are not necessarily the only viable options, but experimentation (shown in subsequent chapters) has shown them to produce advantageous responses. In the next chapter, the performance contributions of the implemented model's individual components will be measured in simulation experiments.

9 Affective Navigation Experiments

Now that the implementation details of the model of robotic affect have been described in Chapter 8, this chapter provides quantitative results that show the performance influences of its various components. This is accomplished by disabling portions of the model and iteratively changing variables controlling the contributions of the components under consideration.

Default affective parameters were provided in Chapter 8. These parameter values are utilised throughout this chapter, except where otherwise stated. Performance effects resulting from variations of these parameters are generally not shown for two reasons. First, the affect model possesses even more tuneable parameters than the underlying planning and control architecture (although many are set to 'obvious' values such as 0 or 1), and there is insufficient time or space to thoroughly test every configuration. Second, it is applied to a relatively well-optimised system, so the performance effects of individual parameters are often very subtle and obscured by noise.

9.1 Constant Drives

Performance characteristics resulting from individual parameter variations were shown in Chapter 7. This section demonstrates the effects of changing these parameters collectively by varying the robot's drive intensities over the range 0 – 1. They are not modulated by the robot's momentary affective states, but maintained at constant values during each experimental run. Any drives that are not the focus of a given experiment are set to their defaults (0.5 for survival drives, 0 for strategic drives).

9.1.1 Safety

The safety drive is tested in environment set C, containing random width doorways and walls. Set C is chosen because it provides a reasonable challenge, but lacks the added complexity of environmental dynamics or undetectable objects. Narrower doorways in these environments can only be traversed if the robot's safety margins and related parameters are relaxed. The completion time (Figure 9.1) shows a small increase until the safety drive intensity $D_{safety} > 0.5$, whereupon it increases rapidly.

Similarly, the success rate (Figure 9.2) remains at 100% until $D_{safety} > 0.5$, and it drops towards 50% thereafter. Increasing the safety drive intensity reduces the likelihood of collisions (Figure 9.3). A significant reduction in collisions occurs while $D_{safety} < 0.3$ and thereafter it decreases at a slower rate. Thus, the optimal value in these environments appears to be around 0.5, but minor improvements to goal convergence or collision likelihood may be achieved by shifting the drive away from this equilibrium position in response to the robot's current situation.

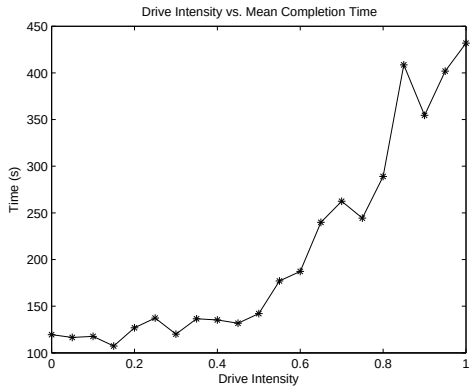

Figure 9.1: Mean completion time for 20 experimental runs per D_{safety} value.

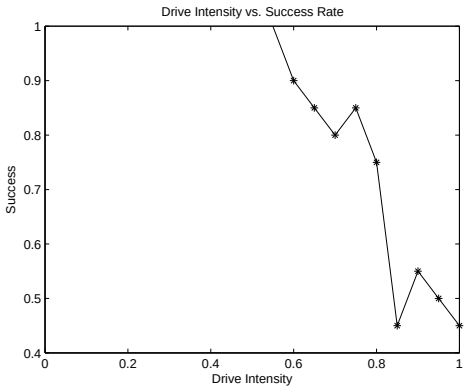

Figure 9.2: Mean success rate vs. D_{safety}.

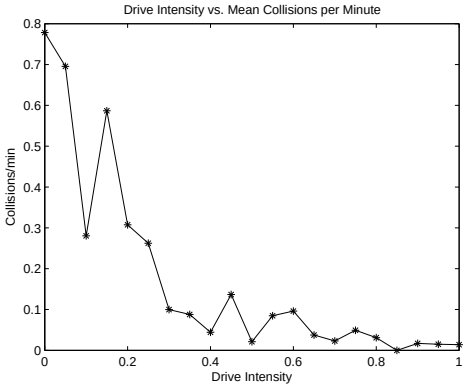

Figure 9.3: Mean collisions per minute vs. D_{safety}.

9.1.2 Speed

The effects of the speed drive on performance are largely dependent on the types of environments in which it is tested. In environment set E (containing dynamic obstacles and random width doorways and walls; representing a significant challenge for high speed navigation) the robot's average velocity (Figure 9.4) increases with

higher speed drive intensities, but this increase begins to level off when $D_{speed} > 0.5$, as environmental obstructions and finite dynamic constraints often prevent the robot from reaching its highest velocities. Initially, the completion time (Figure 9.5) decreases with the speed intensity, but increases when $D_{speed} > 0.5$, as the robot starts to overshoot some doorways rather than turn into them. Collisions occur more frequently at higher speed intensities (Figure 9.6), and any collisions that do occur are likely to cause more damage.

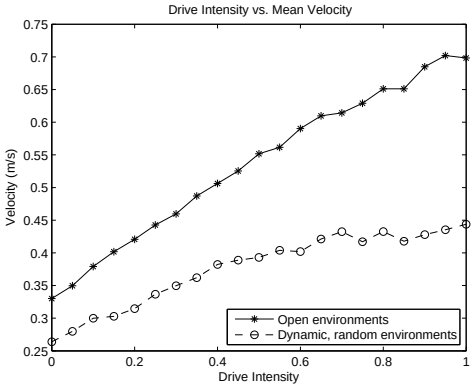

Figure 9.4: Mean velocity for 20 experimental runs per D_{speed} value.

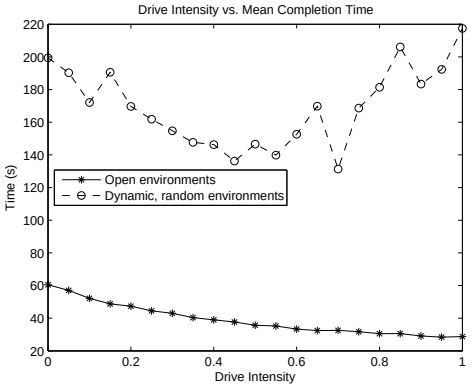

Figure 9.5: Mean completion time vs. D_{speed}.

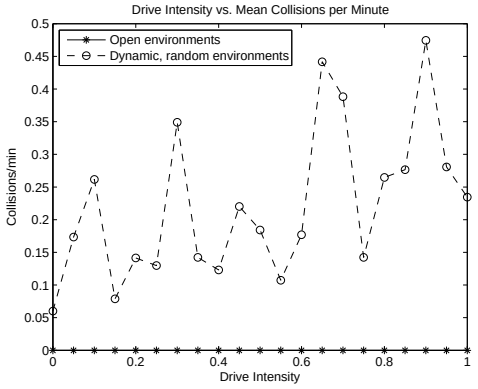

Figure 9.6: Mean collisions per minute vs. D_{speed}.

Conversely, environment set A is sparsely occupied and thus a minimal challenge to navigation (an example is shown below in Section 9.2.1), so there is little incentive to choose lower velocities. The robot's average velocity (Figure 9.4) increases almost linearly with D_{speed}, and higher speeds do not adversely impact completion time (Figure 9.5). No collisions occur regardless of the value of the D_{speed} (Figure 9.6).

This drive is optimal at around 0.5 in environment set E, but in environment set A the highest values (close to 1) are optimal.

9.1.3 Efficiency

Environment set C is again utilised to test the efficiency drive, for the same reason as described for the safety drive. Higher efficiency decreases the execution time ratio (Figure 9.7) due to the reduced quantity of data processed during each control cycle. The lower vector field and obstacle buffer sizes increase the probability that the robot will fail to detect or respond appropriately to an obstacle, resulting in a higher number of collisions (Figure 9.8). This 'less careful' approach also reduces the number of delays and obstructions affecting the robot, reducing its completion time (Figure 9.9) and increasing the success rate (Figure 9.10).

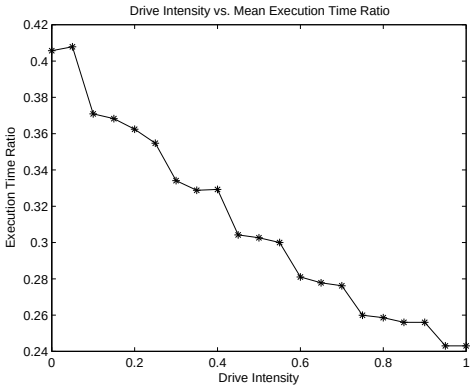

Figure 9.7: Mean execution time ratio for 20 experimental runs per D_{effic} value.

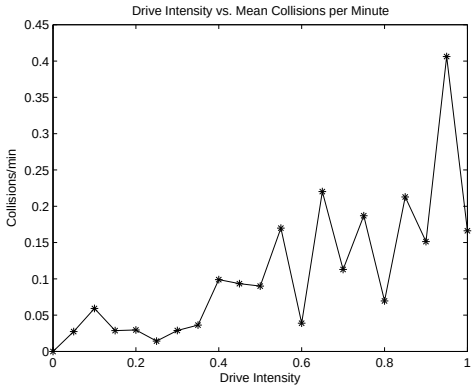

Figure 9.8: Mean collisions per minute vs. D_{effic}.

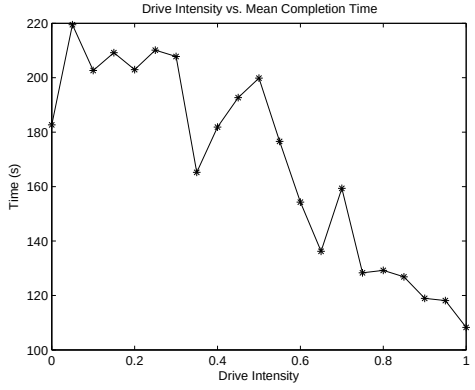

Figure 9.9: Mean completion time vs. D_{effic}.

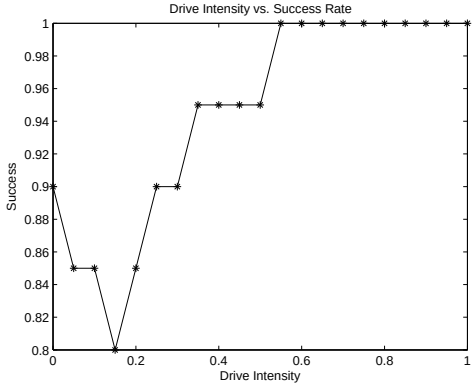

Figure 9.10: Mean success rate vs. D_{effic}.

Overall, the efficiency drive behaves similarly to the safety drive in that it affects the trade-off between goal convergence and collision likelihood, but it also significantly affects processing speed. Unlike the safety drive, the physical performance/safety effects can be regarded as side-effects of changes in computational effort rather than the main purpose of the efficiency drive.

9.1.4 Exploration

In the absence of emotional influences, the effects of increasing the exploration drive are approximately the same as those of increasing the exploration weight directly as shown in Chapter 7. As a result of higher exploration intensities, the robot explores a greater proportion of its environment (Figure 9.11), but this comes at a cost to completion time (Figure 9.12), particularly during early iterations. These results are obtained from three traversals of each environment in set B (chosen because it represents a medium-level challenge) for every datum shown.

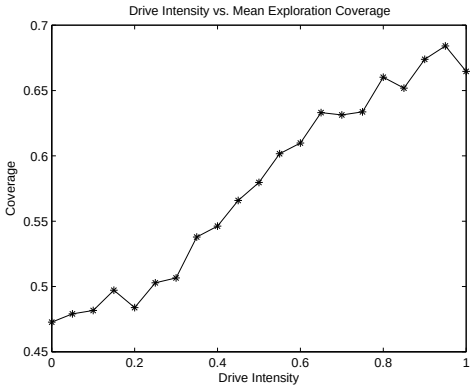

Figure 9.11: Mean coverage for 20 experimental runs per $D_{explore}$ value.

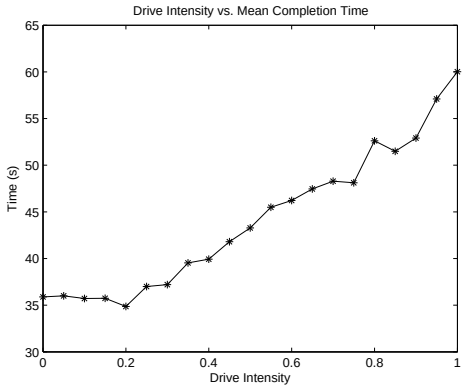

Figure 9.12: Mean completion time vs. $D_{explore}$.

9.1.5 Action

Constant action intensities greater than 0 are of little value to the robot, as it is generally only useful for brief durations while the robot is exploring new areas where its deliberative maps cannot be relied upon. Increasingly random motion produced by the reactive control functions tends to increase completion time (Figure 9.13), and at higher extremes, success rate (Figure 9.14), while most other performance

characteristics remain unchanged. These results are also obtained in environment set B.

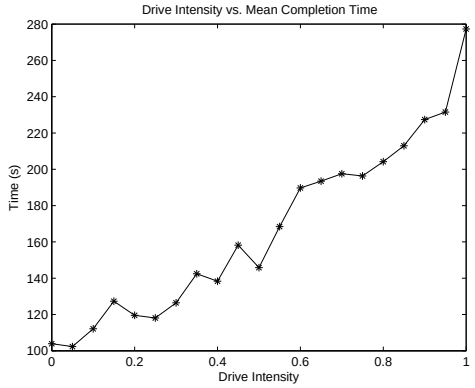

Figure 9.13: Mean completion time for 20 experimental runs per D_{action} value.

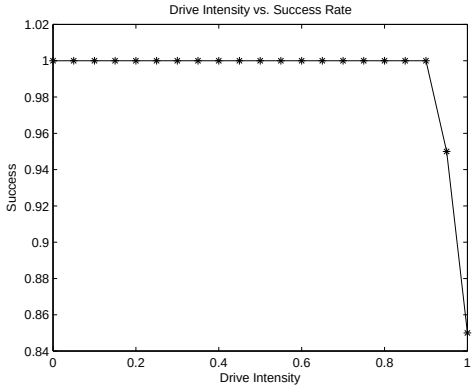

Figure 9.14: Mean success rate vs. D_{action}.

9.1.6 Introspection

Different constant introspection intensities are tested in environment set G, which contains unseen obstacles and random-width doorways and walls. The unseen obstacles are marked on the danger map prior to this experiment. Overall, while higher introspection intensities significantly reduce the likelihood of collisions (Figure 9.15), they also reduce the likelihood that the robot will reach its goal in a timely manner (Figures 9.16 and 9.17). The reasons for this have already been described in Chapter 7.

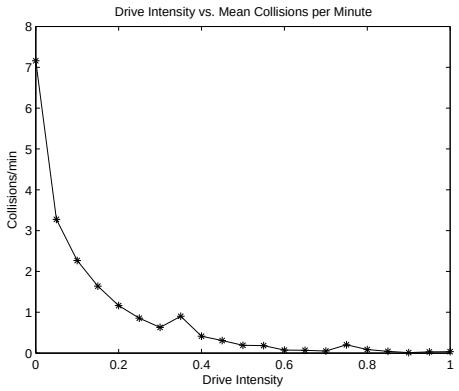

Figure 9.15: Mean collisions per minute for 20 experimental runs per D_{intro} value.

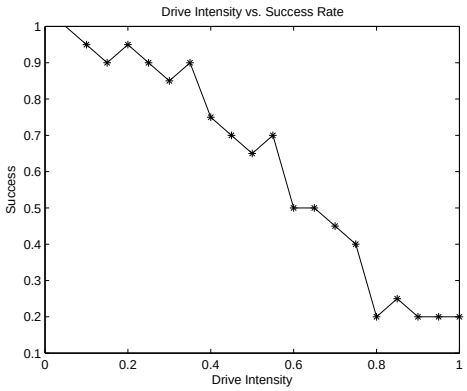

Figure 9.16: Mean success rate vs. D_{intro}.

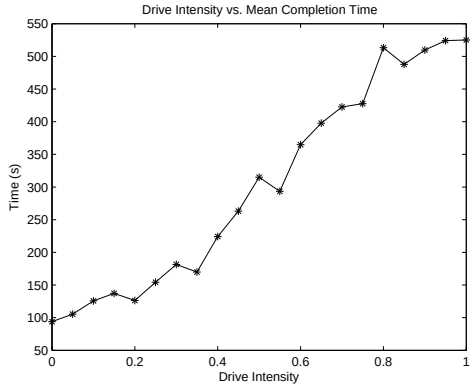

Figure 9.17: Mean completion time vs. D_{intro}.

9.2 Survival Drives, Emotions and Moods

The first set of affective components to be quantitatively analysed are the emotions anger, fear, happiness and sadness and the survival drives (safety, speed and efficiency). These components are interdependent, and their performance characteristics when functioning in isolation do not necessarily reflect those resulting from their integration into the combined system. Hence, these emotions and drives are analysed in combination, while the robot's moods and other affective parameters are iteratively varied.

9.2.1 Global Emotions

Mapped anger, fear, happiness and sadness can be disabled (with respect to parameter modulations) by setting their mapped/global weights φ to 1. This prevents the robot from considering previously elicited emotions E_M when calculating its current emotion intensities E. Instead, they are entirely dependent on the global intensities E_G, functions of currently perceived affective stimuli.

The architecture's performance under varying degrees of emotional activation is measured by manually adjusting its positive and negative moods between 0 and 1. Over this range, the robot's behaviour changes from 'emotionless' (with its survival drives set to constant values of 0.5) to 'bipolar' (where survival drives are modulated within the entire interval [0, 1] depending on the robot's momentary emotional intensities).

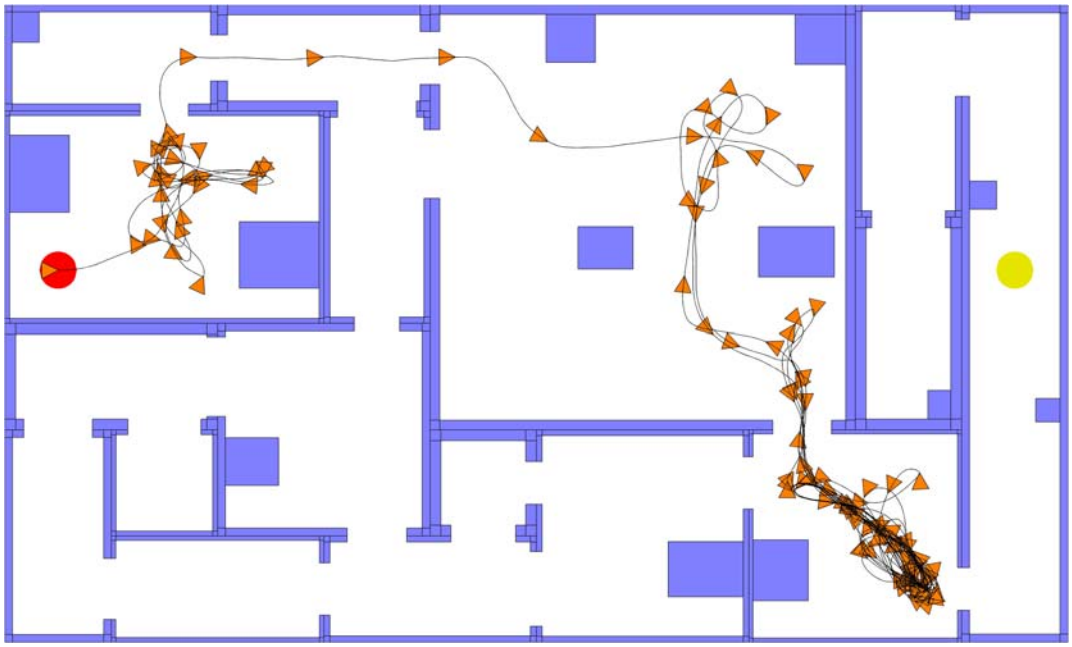

Figure 9.18: Path travelled when positive and negative moods = 0.

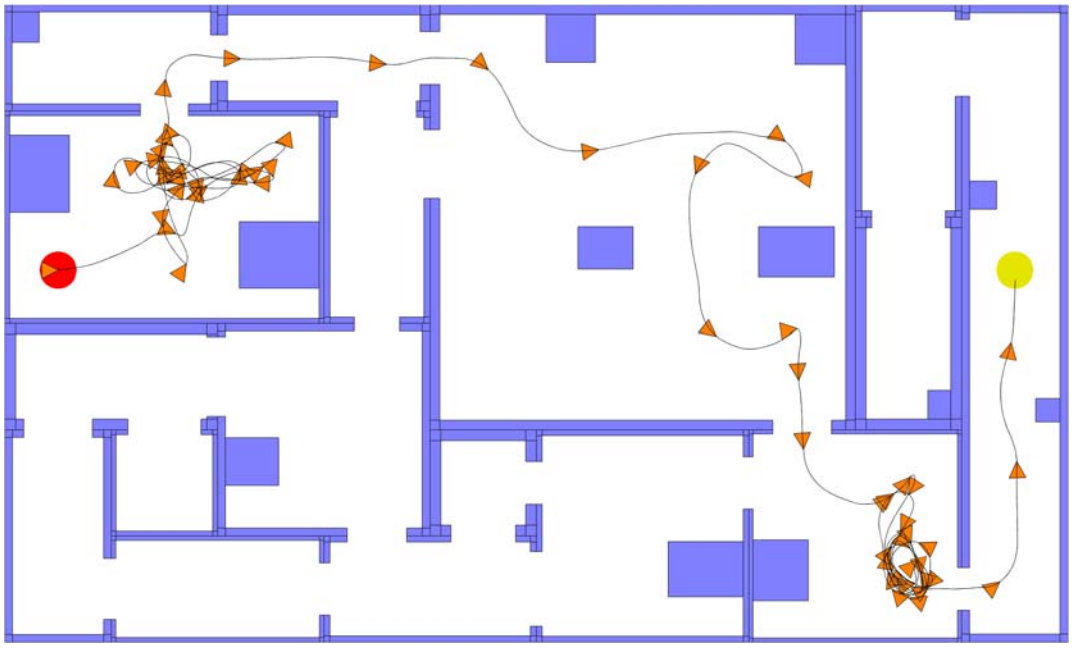

Figure 9.19: Path travelled when positive and negative moods = 1.

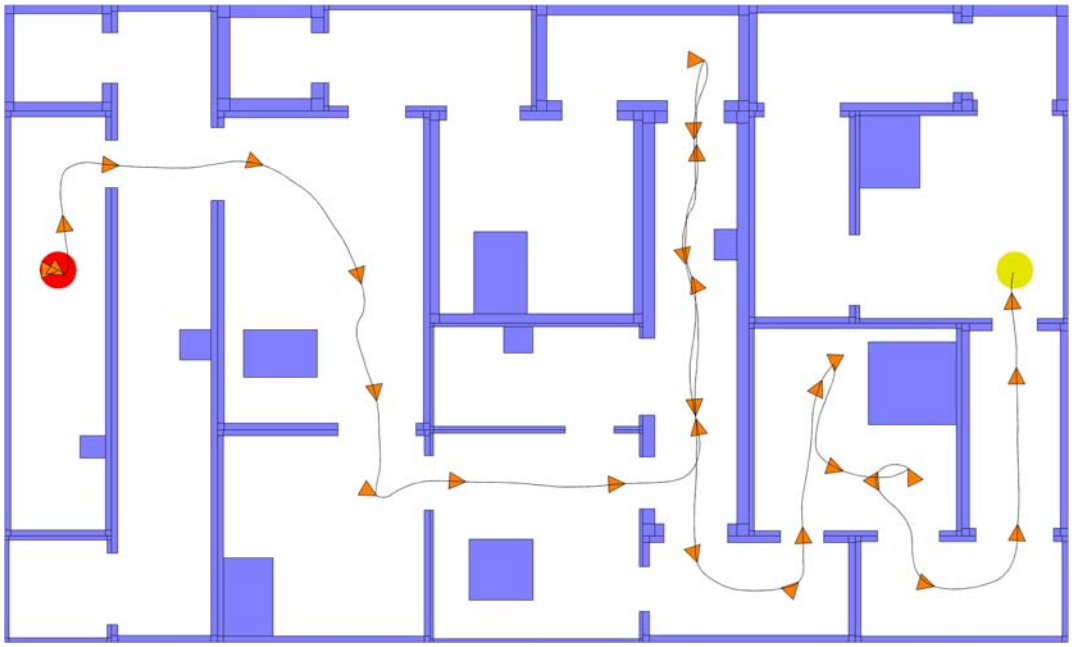

Figure 9.20: Path travelled when positive and negative moods = 0.

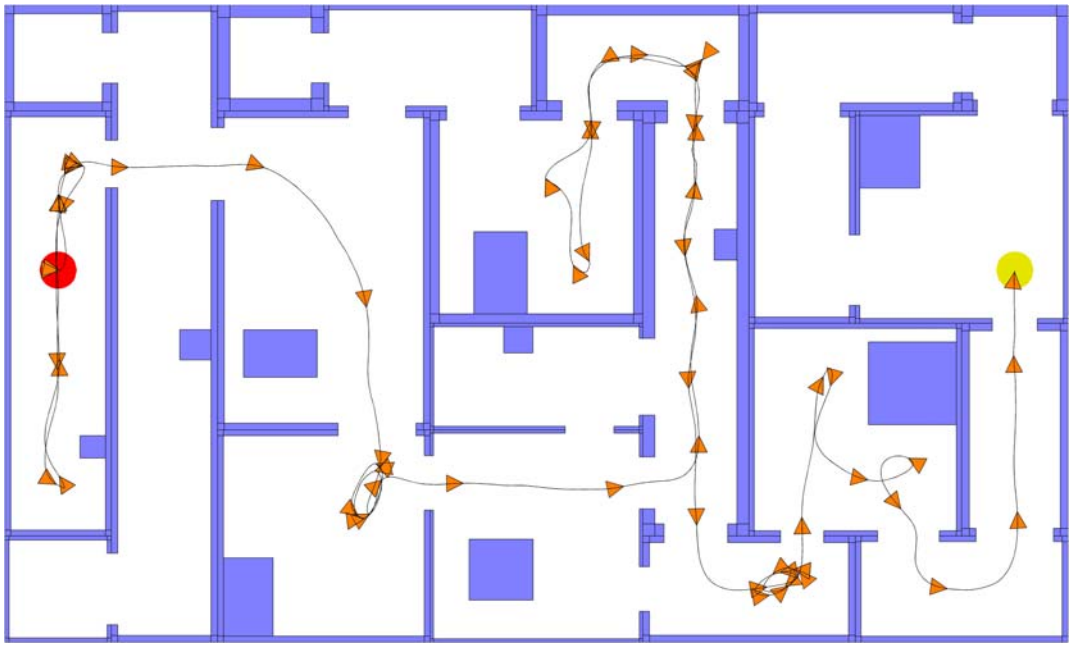

Figure 9.21: Path travelled when positive and negative moods = 1.

This experiment is first conducted in environment set C, containing random width walls and doorways, some of which are only slightly wider than the robot. When the moods are set to low intensities, the robot can usually reach the goal point, although it often takes a long time to traverse the narrowest doorways. Such doorways may require rare combinations of robot position, heading, velocities, and sensor noise to be

perceived as viable openings. Occasionally (e.g. Figure 9.18), these doorways prove to be insurmountable obstructions. With high moods, very few doorways cannot be traversed by the robot, as it tends to reduce the safety drive to its minimal level once its progress has been obstructed for sufficient time to maximise its anger intensity. Under these conditions, the robot is eventually able to traverse the doorway that obstructed the 'emotionless' robot (Figure 9.19). One disadvantage, insofar as global emotions are concerned, is that the 'emotional' robot can become temporarily obstructed by certain doorways that are easily traversed by an 'emotionless' robot with nominally lower safety margins (e.g. Figures 9.20 and 9.21).

Both fear and anger decrease the speed drive. Given the frequency of their elicitation in these challenging environments, the robot's velocities are often lower than those produced in the absence of emotional influences. Hence, the robot's mean velocity decreases as its 'emotionality' increases (Figure 9.22). Coupled with the situational delays (described above), this tends to counteract the reductions in convergence time resulting from the decrease in the number of permanently obstructed states (also described above). Thus, although the success rate (Figure 9.23) grows from around 95% to nearly 100%, the mean completion time (Figure 9.24) remains largely unchanged (barring minor fluctuations due to random noise).

The collision count (Figure 9.25) appears to decrease until the moods are around 0.6, and it increases thereafter. This suggests that higher moods cause the robot to 'overcompensate' for obstructed states. Its safety margins may be lowered to the point where the additional collisions sustained while the robot is 'angry' outweigh the reduction in collisions while it is 'not angry'. In general, these environments can be regarded as a significant navigational challenge, so the robot's happiness intensity tends to be low (and its sadness intensity is high following collisions). This tends to result in lower efficiency, so the execution time ratio increases with mood (Figure 9.26).

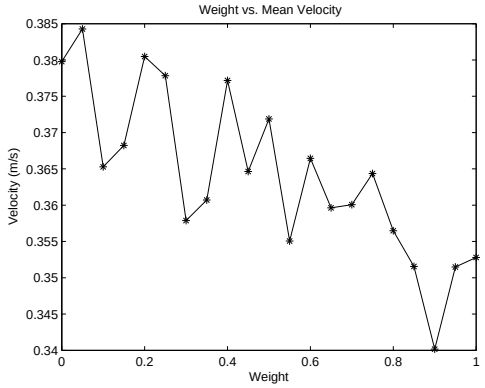

Figure 9.22: Mean velocity for 20 experimental runs per mood intensity.

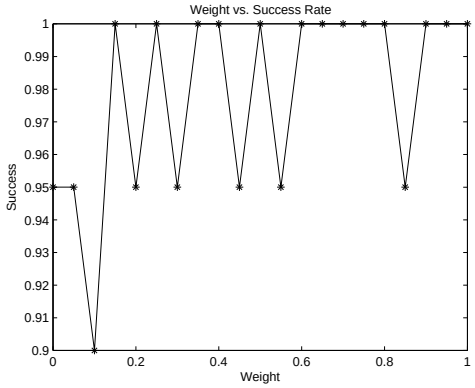

Figure 9.23: Mean success rate vs. mood intensity.

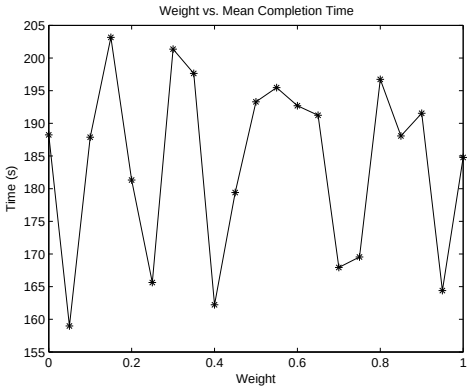

Figure 9.24: Mean completion time vs. mood intensity.

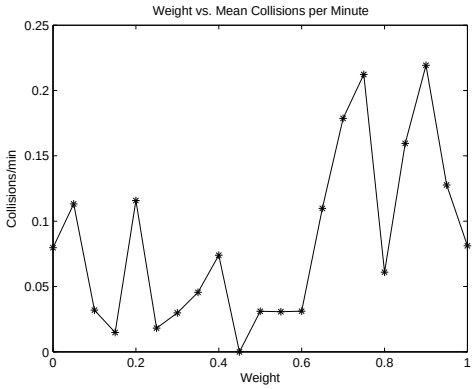

Figure 9.25: Mean collisions per minute vs. mood intensity.

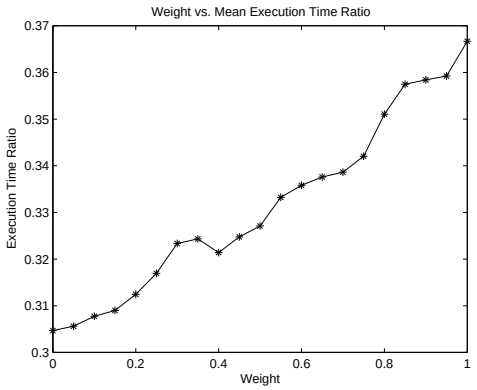

Figure 9.26: Mean execution time ratio vs. mood intensity.

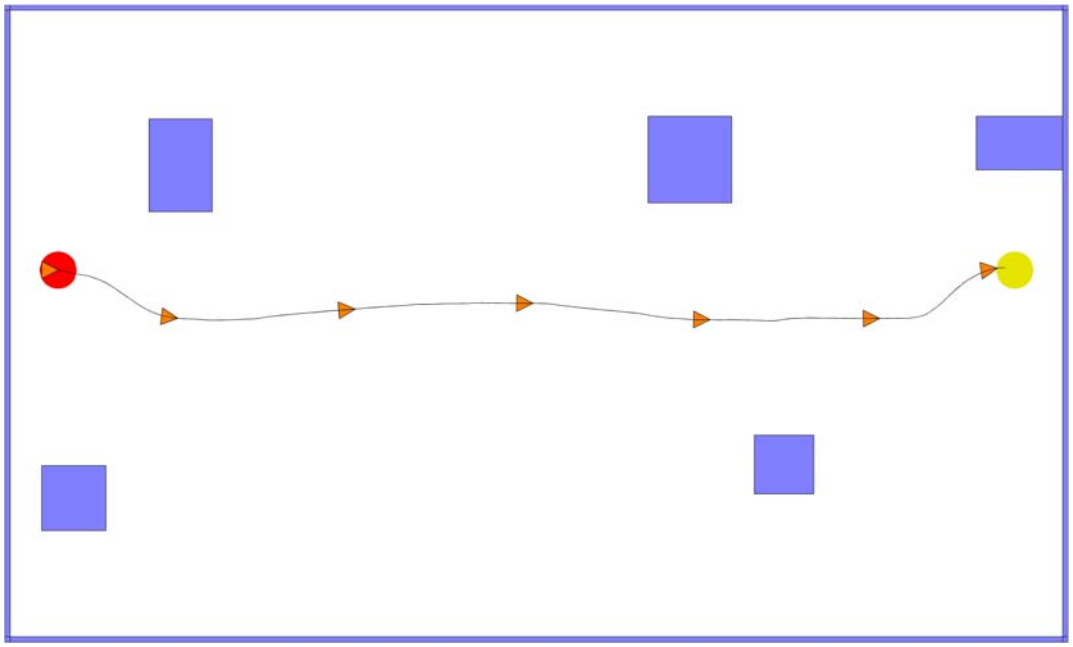

Figure 9.27: Path travelled when positive and negative moods = 0.

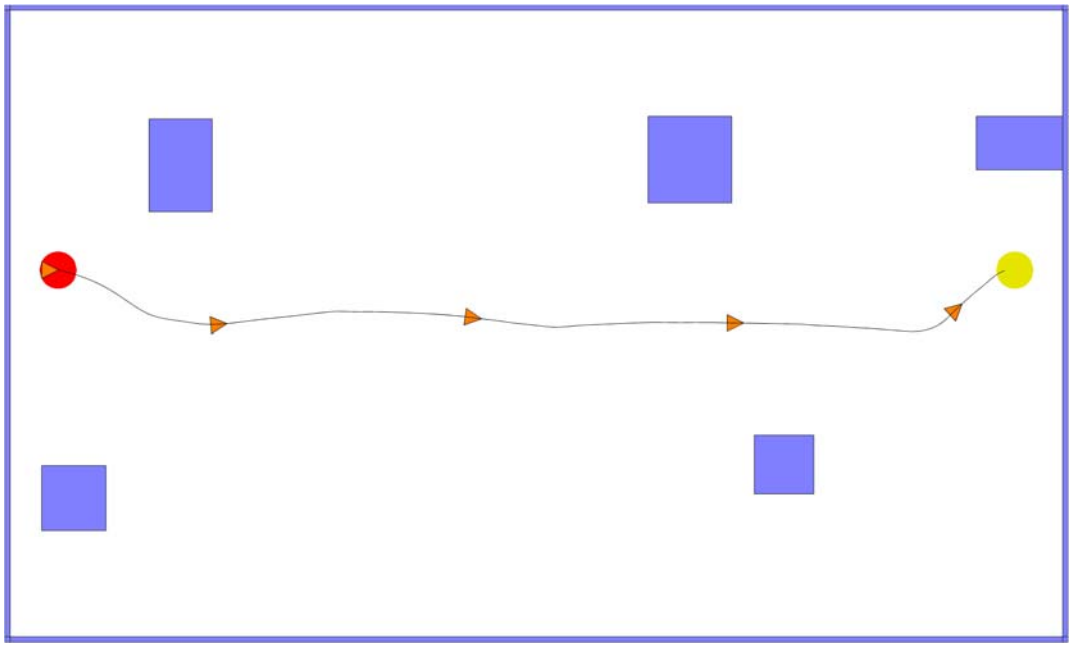

Figure 9.28: Path travelled when positive and negative moods = 1.

The experiment is repeated in environment set A, representing open environments with sparsely distributed obstacles (Figures 9.27 and 9.28). These environments are very unchallenging; collisions do not occur, and the robot can easily find a clear path to the goal. Anger, fear and sadness intensities tend to be very low (and usually 0), while happiness quickly converges on 1 after the robot begins moving, due to its

unobstructed path to the goal. Correspondingly, speed, safety and efficiency tend towards their maximum intensities.

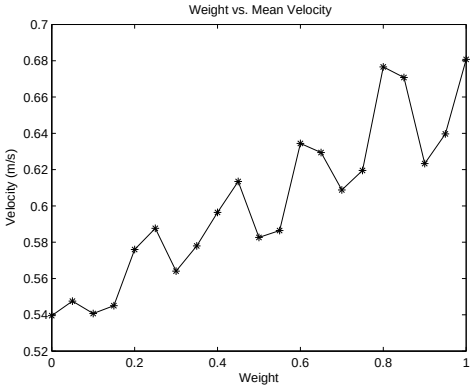

Figure 9.29: Mean velocity for 20 experimental runs per mood intensity.

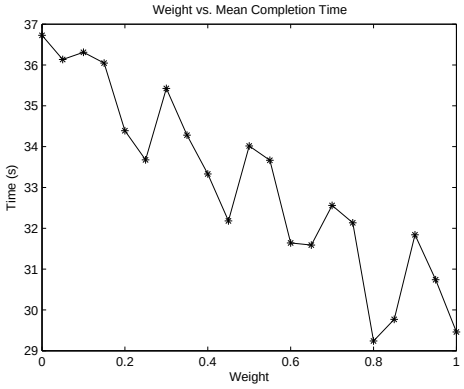

Figure 9.30: Mean completion time vs. mood intensity.

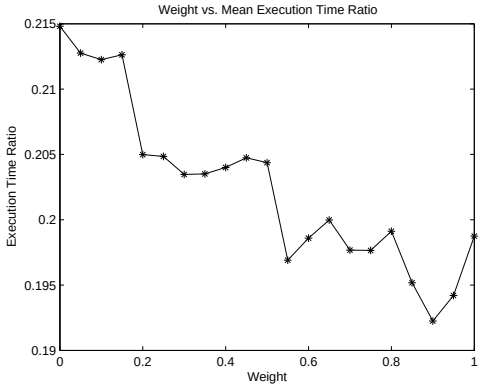

Figure 9.31: Mean execution time ratio vs. mood intensity.

Increasing the mood intensities has a universally positive effect on performance in these environments. The robot's mean velocity increases due to the higher speed drive (Figure 9.29), while the completion time decreases (Figure 9.30). The execution time ratio decreases due to the higher efficiency drive (Figure 9.31). Over the entire range of mood values, the success rate and collision count remain at 100% and 0, respectively, indicating that the value of the safety drive is largely irrelevant in such simple environments.

The results obtained in these two environment sets show that the tested global emotions and drives can alter the robot's response to different environmental conditions in a manner that is beneficial to its overall performance. Performance can

be further improved by allowing mapped emotions to exert an influence over these responses.

9.2.2 Mapped Emotions

When utilised in isolation, global emotions are limited by the absence of long-term memory. This is particularly apparent for our implementation of anger. By temporarily lowering the safety drive, global anger can enable the robot to overcome most obstructions. To prevent 'false positive' incidences of anger that may unnecessarily jeopardise the robot's ability to avoid collisions, the growth rate of anger is relatively slow (and the decay rate is very fast). An undesirable consequence of this slow growth is that it takes some time for the safety drive to lower sufficiently to allow the robot to traverse narrow doorways. Regardless of the number of times it encounters the same doorway, the robot must wait for its anger to build before it can traverse it. For example, in Figure 9.32, the robot becomes obstructed by a doorway during its fifth pass through the environment (the previous four traversals are omitted from the figure, as they would otherwise obscure the fifth one).

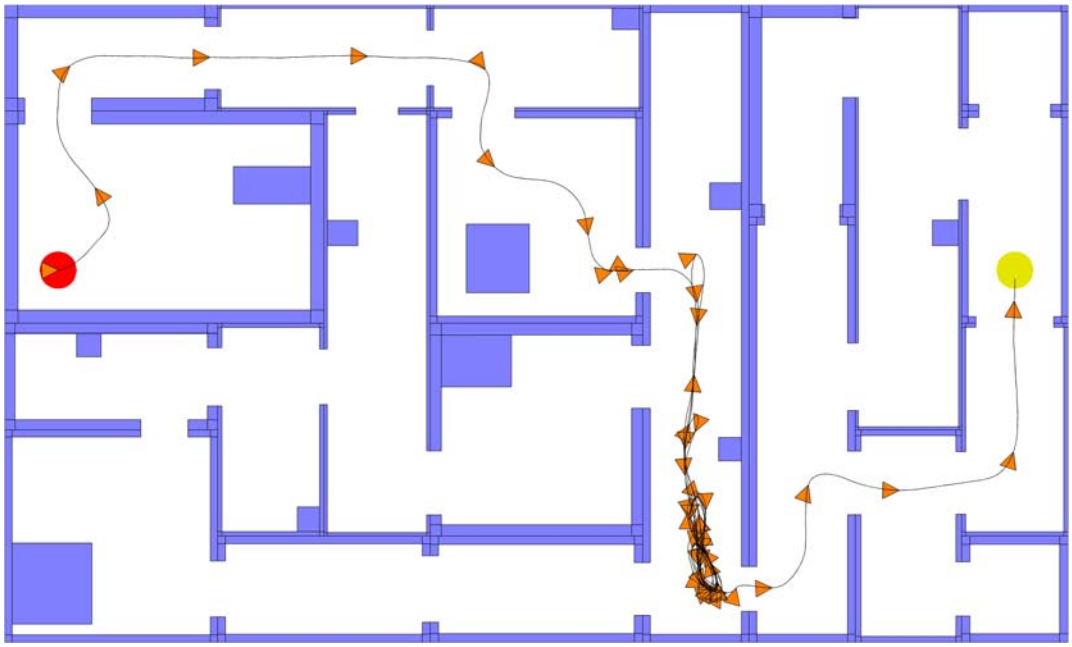

Figure 9.32: Path travelled when $\varphi = 1$.

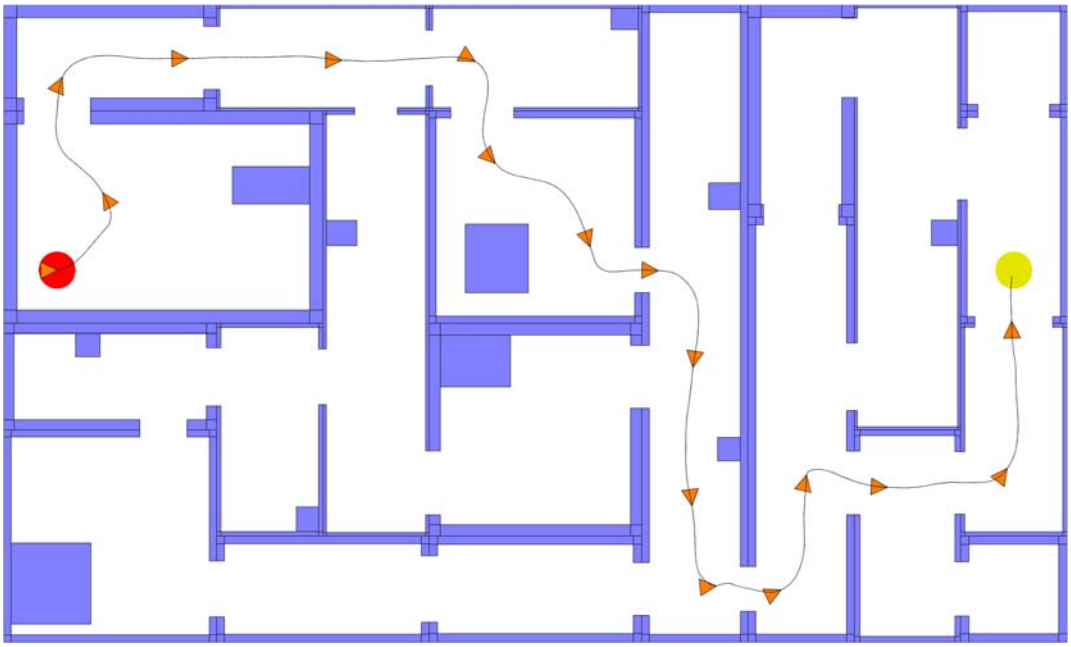

Figure 9.33: Path travelled when $\varphi = 0$.

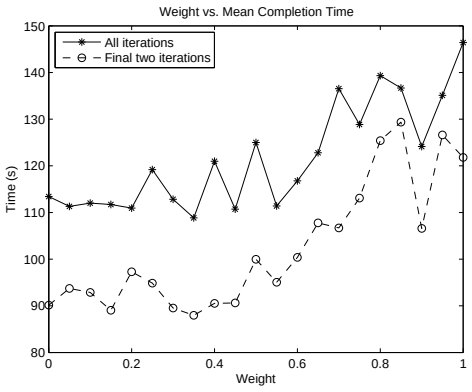

Figure 9.34: Mean completion time for 20 experimental runs per φ value.

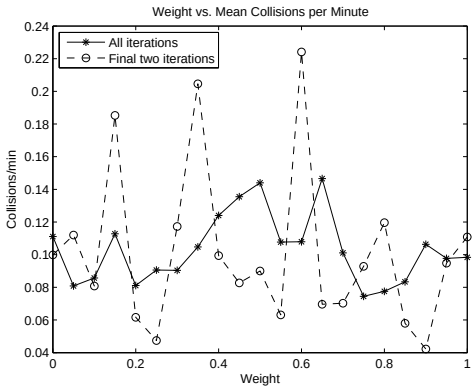

Figure 9.35: Mean collisions per minute vs. φ.

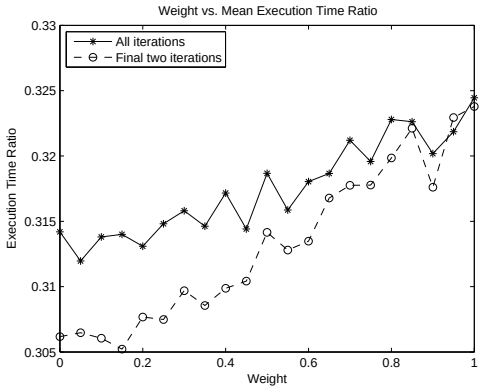

Figure 9.36: Mean execution time ratio vs. φ.

Mapped emotions allow the robot to instantly adapt its emotion intensities to match those that were previously elicited near its current location. For anger, this means that the robot's safety drive rapidly lowers as it approaches a doorway that previously obstructed its progress, allowing it to traverse the doorway without delay. Figure 9.33 shows the robot's fifth pass through an environment when it utilises mapped emotions exclusively. The doorway that delayed the robot's progress when it was reliant on global emotions is no longer a significant obstruction.

Various performance characteristics are measured while the weight φ controlling the relative contributions of mapped and global emotions is varied between 0 (fully mapped) and 1 (fully global). Positive and negative moods are both kept constant at 0.5, a value that allowed reasonably significant modulations in the previous experiments without decreasing the collision count. The completion time (Figure 9.34) is noticeably lower when the robot is more heavily reliant on mapped emotions, primarily due to the decreased delays associated with anger. This effect is more noticeable during later iterations, once the mapped emotions have been updated to suit the environments. Improvements to completion time comes at no apparent cost to the collision count (Figure 9.35), which remains largely constant, aside from a slight peak at the middle of the range that can likely be attributed to noise.

A decrease in the execution time ratio (Figure 9.36) occurs if mapped emotions are highly enabled, indicating higher computational efficiency, particularly during later iterations. This is caused by the higher levels of happiness resulting from faster goal convergence. Higher efficiency may be undesirable, however, as it generally comes at a cost to safety and performance in challenging environments. There remains a high risk of collisions in these environments even after the robot's convergence speed increases. Hence, although anger clearly benefits from low φ values, it may be beneficial for the equivalent parameter for happiness to have a higher value.

Mapped emotions can also be utilised as planning biases, increasing or decreasing the costs associated with their nodes. Experiments in three different environment sets (A, B and G) are conducted while the overall contribution of mapped emotions to path planning is varied by iteratively increasing the weight W_{p4}.

The resulting effects on performance are heavily dependent on the type of environment in which the experiment is conducted. The example shown in Figures

9.37 and 9.38 represents a relatively challenging environment with numerous narrow doorways. As a result, the robot's happiness intensity tends to be low (due to its slow overall progress), while anger is frequently high (Figure 9.39). This results in a strong negative bias to many nodes through which the robot has previously passed. Ideally, this encourages the robot to search for alternative paths that may be more suitable. However, in this example the alternative paths are not superior to those previously travelled, so the robot simply behaves as if a high exploration bias has been applied to path planning. Conversely, in the environment shown in Figure 9.40, the robot has no difficulty converging on the goal point, so its happiness intensity is generally high (Figure 9.41) and anger low. Thus, the robot's initial path is deemed successful, so it adheres to it thereafter.

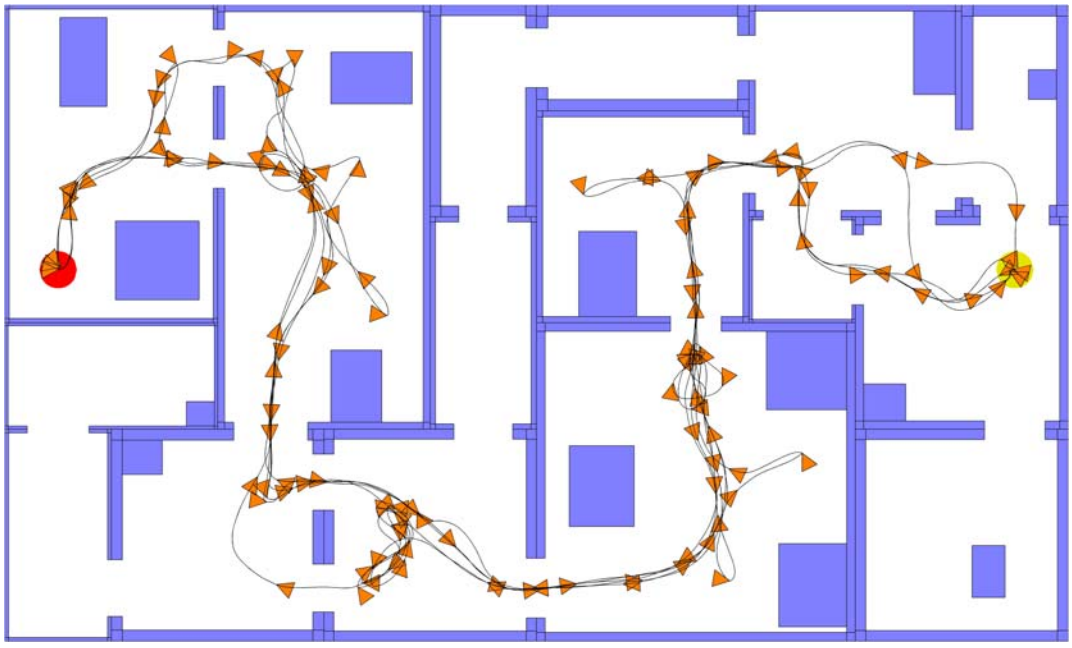

Figure 9.37: Path travelled when W_{p4} = 0.

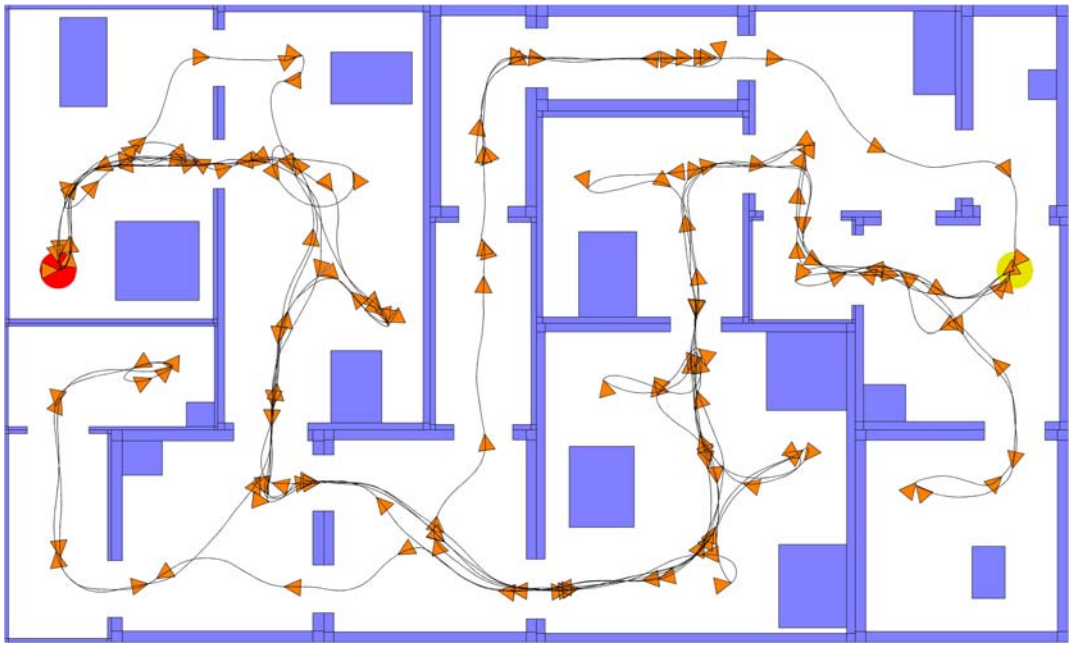

Figure 9.38: Path travelled when W_{p4} = 0.5.

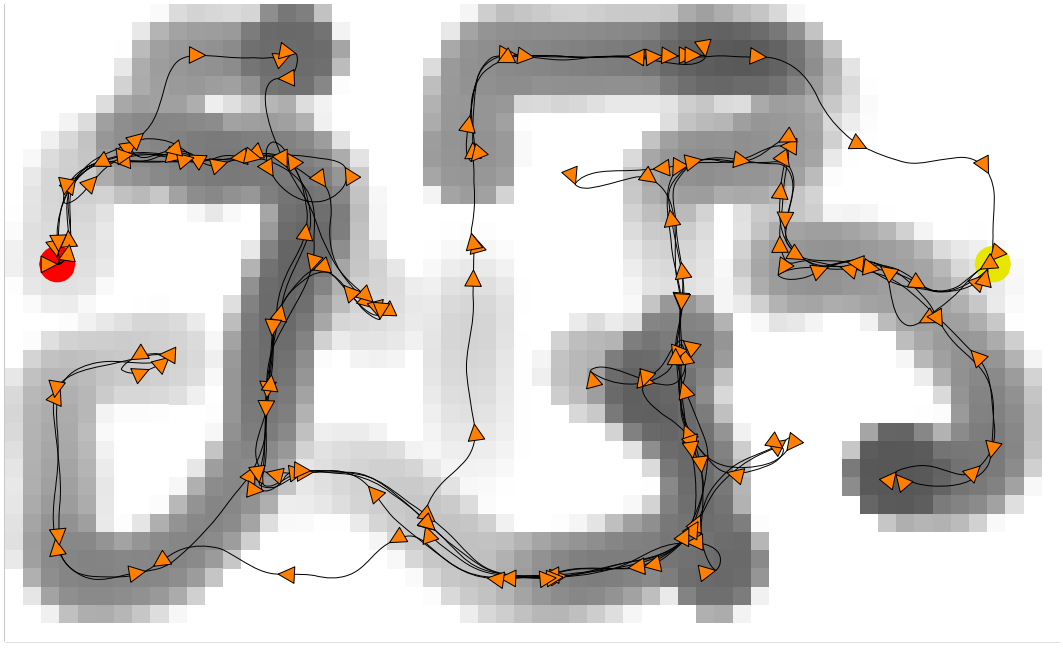

Figure 9.39: Corresponding anger map.

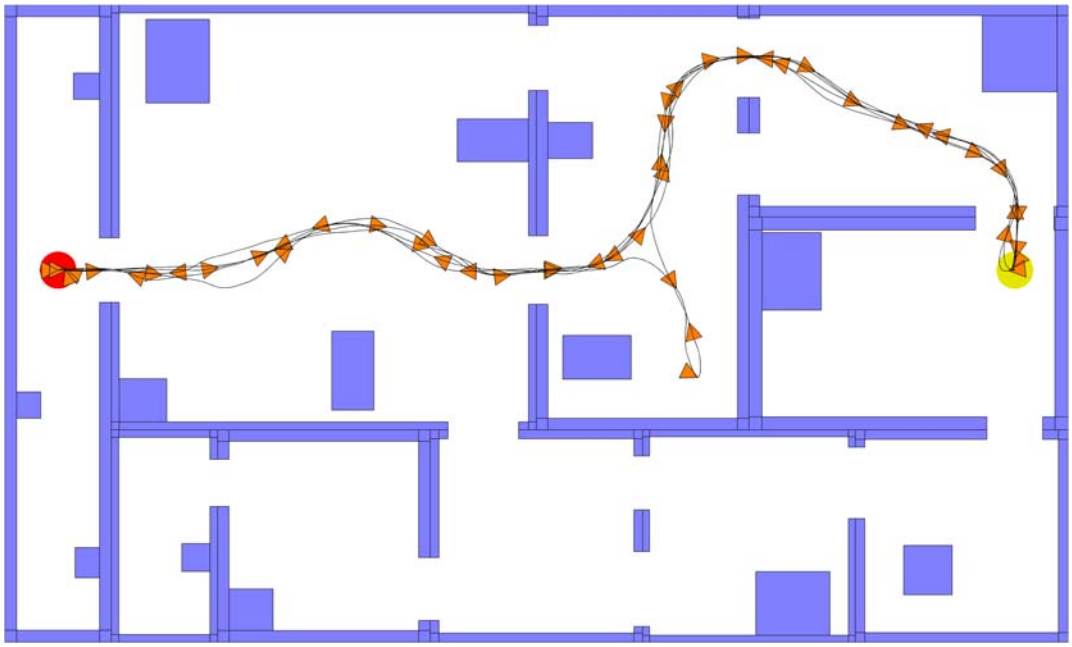

Figure 9.40: Path travelled when W_{p4} = 0.5.

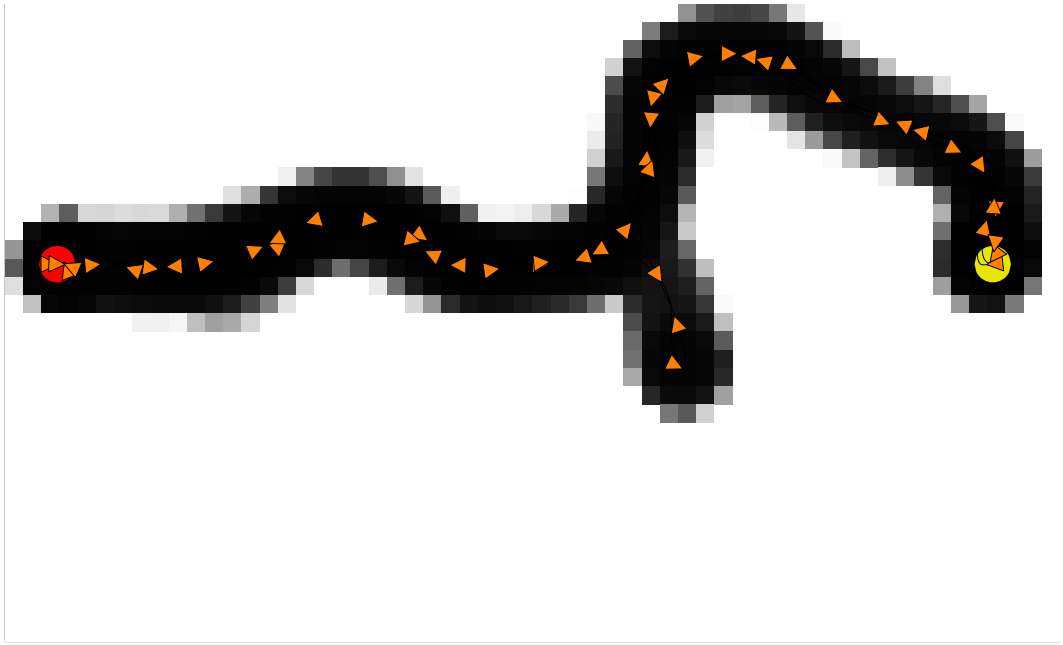

Figure 9.41: Corresponding happiness map.

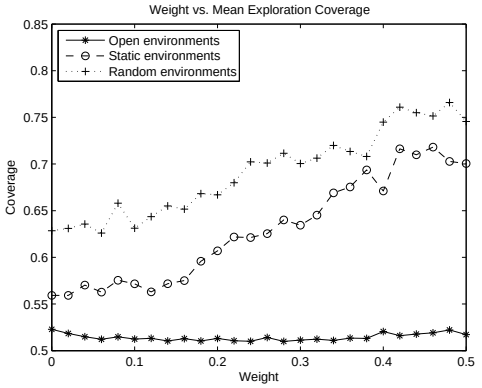

Figure 9.42: Mean exploration coverage for 20 experimental runs per W_{p4} value.

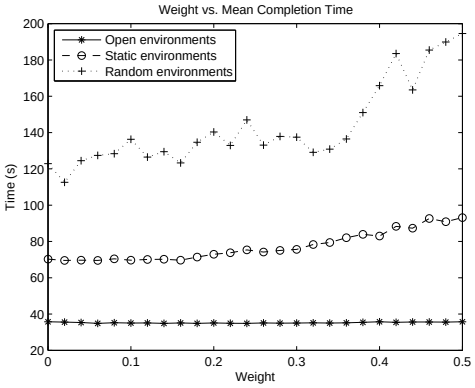

Figure 9.43: Mean completion time vs. W_{p4}.

Overall, the mapped emotions yield increased coverage of simulated office-block environments (sets A and B), but this comes at a cost to completion time due to the less direct paths being chosen (Figures 9.42 and 9.43). In open environments (set A), the exploration coverage and completion time remain largely unchanged over the W_{p4} range, because the first path travelled yields maximal happiness and minimal anger and sadness, and thus the robot has little incentive to seek alternative paths.

9.2.3 Moods

The effects of dynamic moods on performance are measured by iteratively increasing a weight limiting (or multiplying) their intensities. If this weight is set to 1, mood intensities rarely exceed 0.5. Thus, the weight is varied over the range 0 – 2 so that moods can occupy the intended unit interval (but they are not allowed to exceed 1). In this experiment the contribution of mapped emotions to parameter modulations are enabled, but their contributions to path planning are disabled.

A slight decrease in the average completion time (Figure 9.44) occurs as the moods are more highly enabled, and this decrease is more substantial during the final two iterations once the emotion maps have been updated. This is accompanied by a minor increase in success rate from 96% to around 98% (Figure 9.45), and an apparent increase in the number of collisions per minute (Figure 9.46). The execution time ratio (Figure 9.47) also increases, but the level of growth is lower in later iterations, where the robot encounters fewer obstructions and thus possesses greater happiness and positive mood intensities.

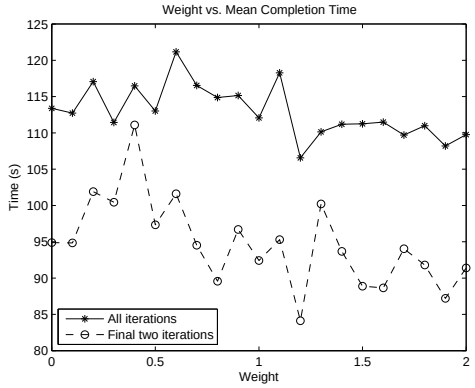

Figure 9.44: Mean completion time for 20 experimental runs per mood weight.

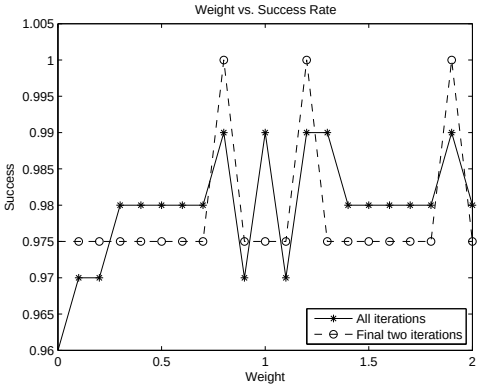

Figure 9.45: Mean success rate vs. mood weight.

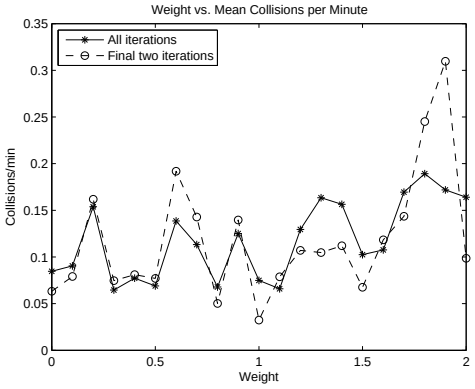

Figure 9.46: Mean collisions per minute vs. mood weight.

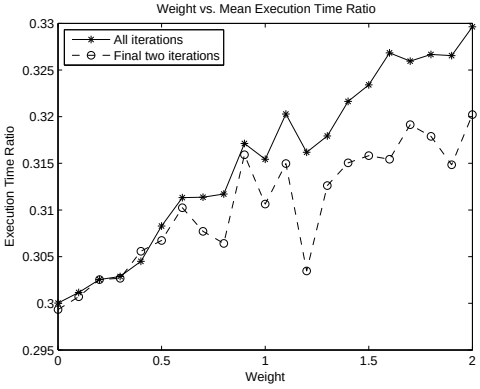

Figure 9.47: Mean execution time ratio vs. mood weight.

Next, the mood weights are set to constant values of 1 (a compromise between goal convergence and collision likelihood), and W_{p4} is varied over the range 0 – 1. Unlike in the previous experiments with mapped emotions, mood intensities modulate the contributions of mapped emotions to path planning. Positive mood controls the contribution of happiness, while negative mood does the same for sadness, fear and anger. The cost stimulus also regulates the contributions of moods to planning so that emotional influences are only utilised when they are unlikely to be detrimental to performance.

With static moods, higher W_{p4} values in environment set C yielded greater exploration coverage, but this came at a significant cost to completion time. When emotional influences to path planning are regulated by moods and the cost stimulus, this cost is significantly reduced. The system becomes self-limiting; high W_{p4} values often produce paths that result in a high cost stimulus, and the stimulus subsequently

reduces the emotional contribution, lowering the incentive to find superior paths. Hence, the exploration coverage (Figure 9.48) reaches a maximum of around 0.71 when $W_{p4} \approx 0.5$, and remains largely constant thereafter. Average completion time (Figure 9.49) is relatively constant, with a minor increase over the middle of the range that could be attributed to noise. The completion time recorded during the final two iterations (once the environments have been largely explored) shows a possible minor decrease, although the large fluctuations make this potential trend unclear.

When properly regulated, emotional influences to path planning appear to have a generally beneficial influence on performance. However the effect of a given W_{p4} value is likely to be different when it is utilised in combination with the exploration map (whose contribution to planning is controlled by the exploration drive, demonstrated in Section 9.3.1). In sparsely occupied environments (e.g. set A) the two influences are expected to partially cancel each other out, whereas in challenging environments (e.g. set C) they are likely to be additive. Quantitative effects of the combined system are shown in Chapter 10.

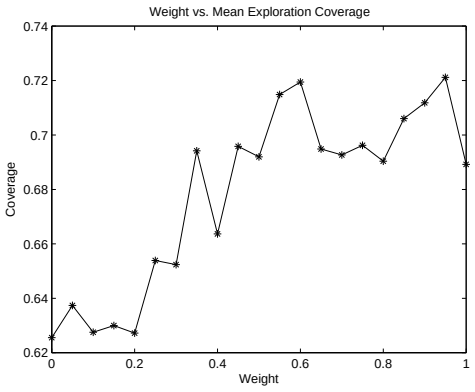

Figure 9.48: Mean exploration coverage for 20 experimental runs per W_{p4} value.

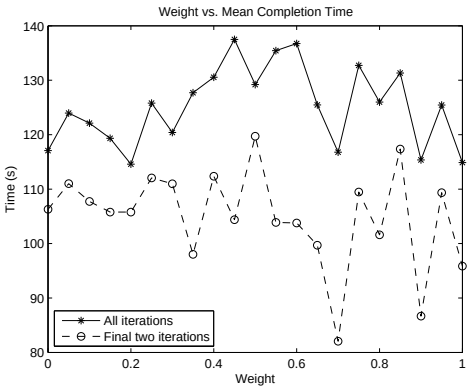

Figure 9.49: Mean completion time vs. W_{p4}.

9.3 Strategic Drives and Emotions

Next, the performance contributions of the three remaining emotions (curiosity, surprise and confusion) and the strategic drives (exploration, action and introspection) are measured. Because of their reduced interdependencies, these components can be more easily decoupled from the rest of the model to analyse their performance

contributions independently. They are not entirely independent from the other emotions and drives, however (e.g. surprise heavily influences the safety drive).

9.3.1 Curiosity, Surprise and Exploration

The exploration drive is primarily modulated by curiosity and surprise, so an experiment is conducted with these emotions enabled while varying the maximum exploration intensity. Happiness and sadness, which normally exert small influences over exploration, are disabled (set to 0) so that the effects of the primary influences (curiosity and surprise) can be analysed independently. Environment set B is utilised for this experiment, and the robot is instructed to travel back and forth through each environment for a total of five iterations.

With exploration disabled (Figure 9.50), the robot strongly favours paths previously travelled. When the weight controlling its maximum intensity is set to 1 (Figure 9.51), the robot initially favours exploration over following the shortest path, so a large proportion of the environment is covered during early iterations. Curiosity (the dominant influence) is self-limiting (due to its cost stimulus); it is only allowed to achieve its maximum intensity during early iterations, and only when the resulting planned paths are viable. Once the environment has been largely explored, the rate of knowledge acquisition (learning stimulus) slows, causing curiosity and exploration to subside.

Overall, the proportion of the environment explored grows as the weight controlling the maximum exploration intensity increases (Figure 9.52). This comes at a cost to the completion time averaged over all iterations (Figure 9.53), due to the longer paths chosen during early iterations. However, if we consider only the final two iterations (after the majority of environmental learning is completed) this cost is greatly reduced.

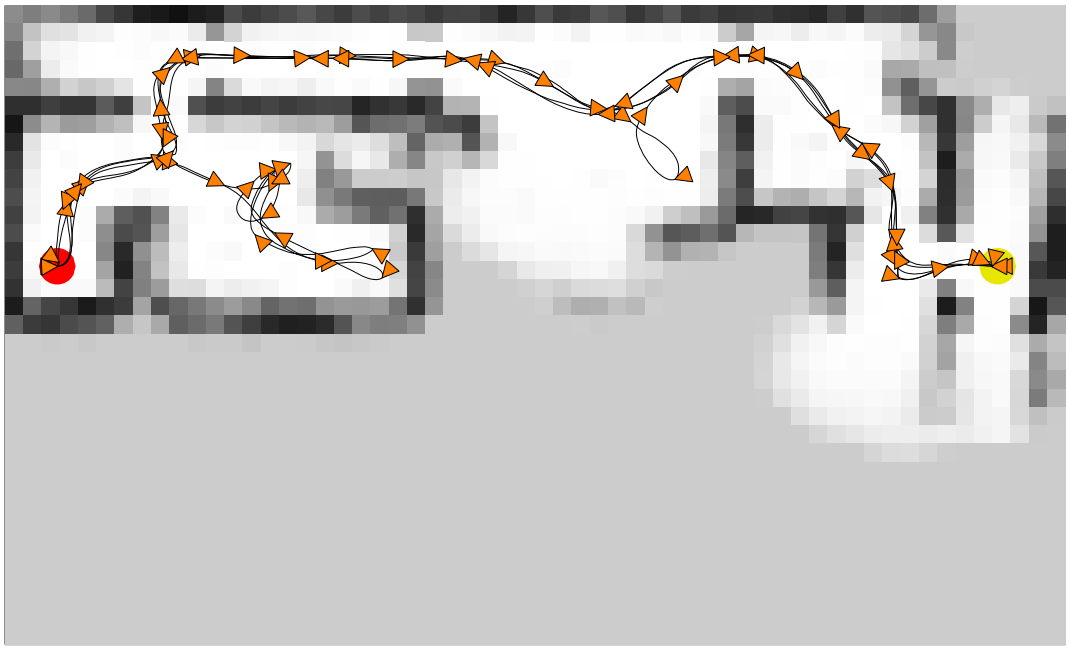

Figure 9.50: Path travelled and occupancy grid map when the exploration weight = 0.

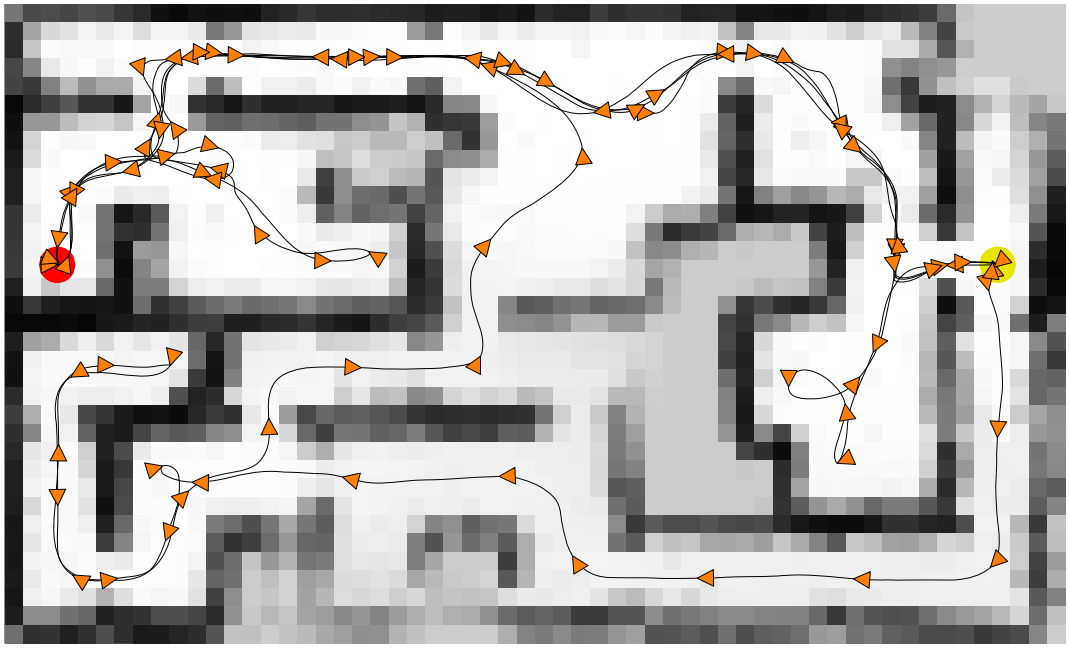

Figure 9.51: Path travelled and occupancy grid map when the exploration weight = 1.

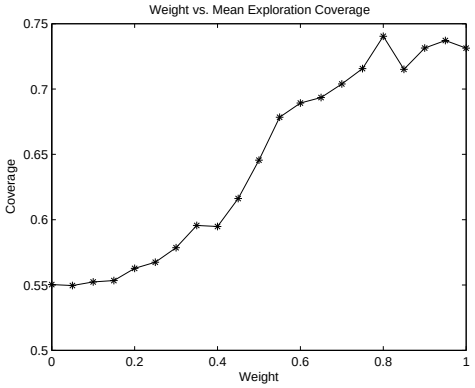

Figure 9.52: Mean coverage for 20 experimental runs per exploration weight.

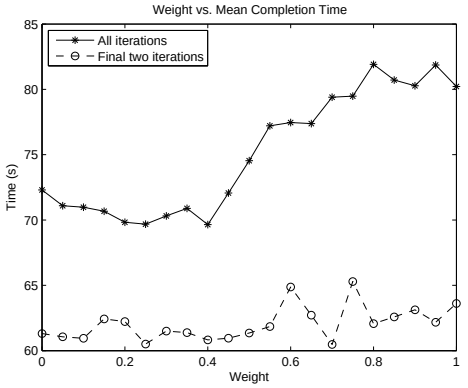

Figure 9.53: Mean completion time vs. exploration weight.

9.3.2 Surprise, Action and Safety

The emotion surprise and its connected drives (action and safety) are tested by varying a weight controlling the influence of these drives in environment set B. Although confusion can normally influence the action drive, there are no undetectable objects or sensor failures to elicit it in these environments. Anger and fear are set to constant values of 0.5 and 0, respectively, so that surprise is the only source of modulations for the safety drive, and these modulations remain centred around 0.5. This experiment is conducted once under normal conditions, then with the robot's dynamic mapping capabilities disabled and an incorrect map represented.

With mapping enabled and functioning correctly, there are few advantages to increasing the robot's reliance on reactive navigation strategies, so no tangible performance improvements are apparent. However, the decrease in performance that resulted from simultaneously enabled reactive and deliberative navigation strategies previously shown in Section 6.4 is eliminated. Instead, the mean completion time (Figure 9.54) and success rate (Figure 9.55) remain largely constant regardless of the strength of action drive modulations. This is because the contributions of purely reactive control inputs are only temporarily activated while the robot explores new locations. Once it has the world knowledge required to produce satisfactory planned paths, they are effectively disabled.

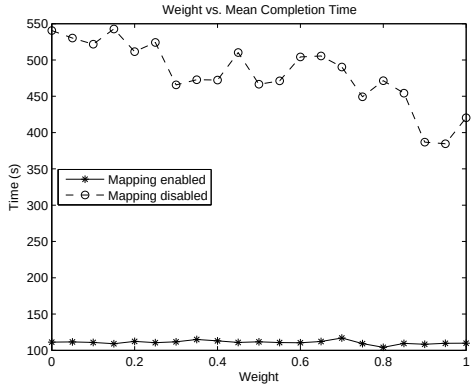

Figure 9.54: Mean completion time for 20 experimental runs per action weight.

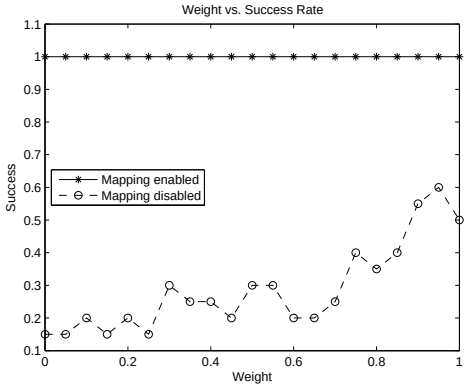

Figure 9.55: Mean success rate vs. action weight.

The advantages of surprise only become apparent in the second experiment, where the robot's occupancy grid map incorrectly represents the entire environment as free space. The mean completion time decreases as the contribution of the action drive increases (Figure 9.54), and the success rate tends towards a maximum of around 60% (Figure 9.55). With this contribution entirely disabled, the robot is reliant on the path following function. Due to the empty grid map, planned paths often pass through walls, so the robot is highly susceptible to becoming trapped by local minima (Figure 9.56). When the action drive is sufficiently enabled for reactive navigation to become the dominant strategy, the robot can often escape from these local minima due to the increased contributions of the angular inertia and wander functions (Figure 9.57).

If the safety drive is set to 0.5 in environment set B, the collision rate is essentially zero when the robot employs a hybrid reactive/deliberative navigation strategy. However, purely reactive navigation responds poorly to such low safety margins. This is best illustrated by the example shown in Figure 9.58, where the robot enters the gap between the wall and the object in the lower right corner of the environment. The gap is little wider than the robot, so it sustains numerous collisions and becomes trapped in the resulting local minimum. When the safety drive is modulated by surprise (and deliberative mapping is disabled), the resulting safety margins are often high enough to dissuade the robot from attempting to enter such a narrow opening (Figure 9.59).

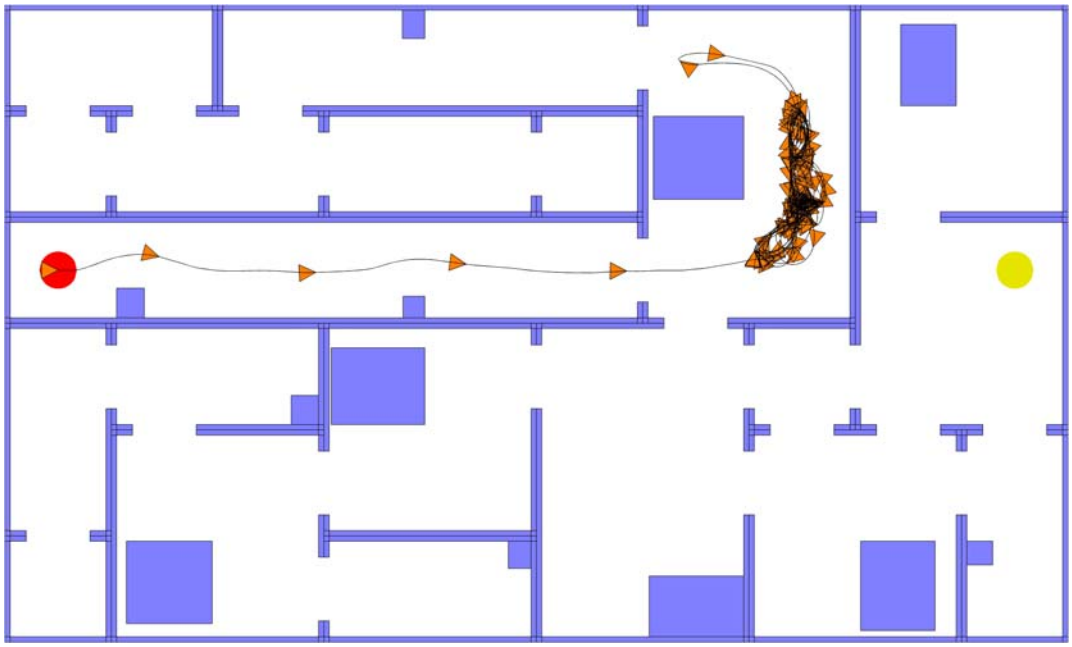

Figure 9.56: Path travelled when the action weight = 0.

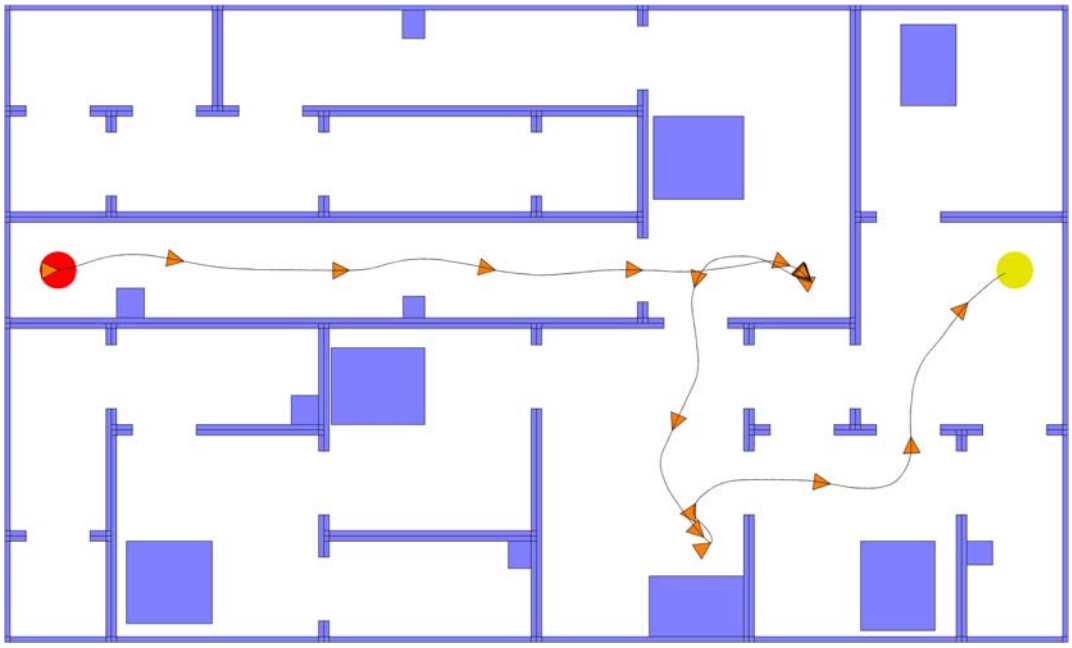

Figure 9.57: Path travelled when the action weight = 1.

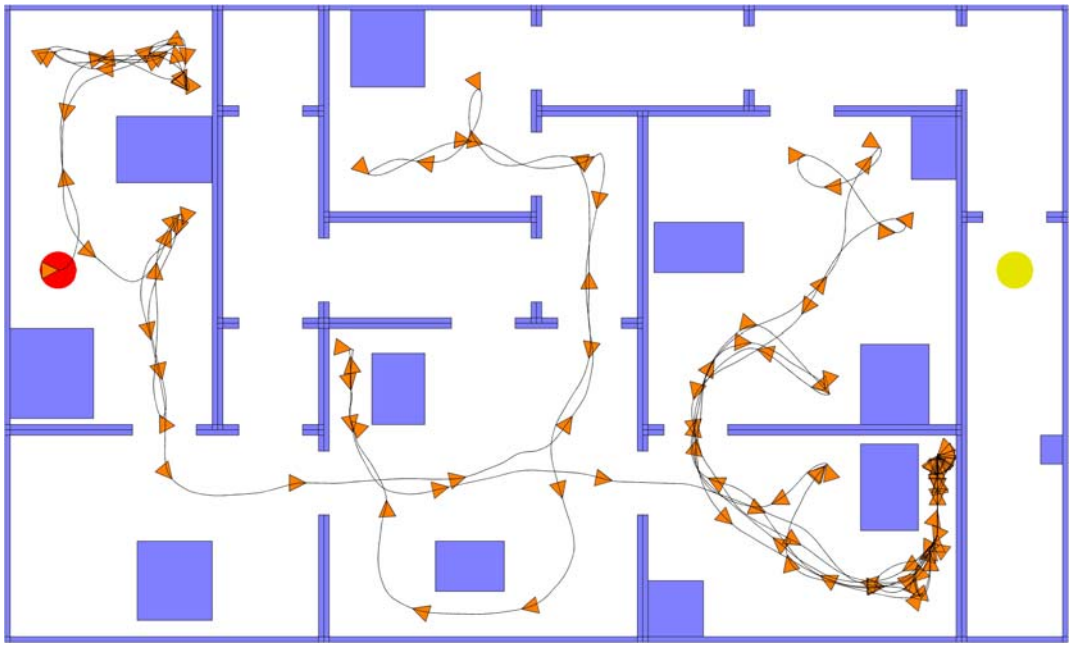

Figure 9.58: Path travelled when the safety weight = 0.

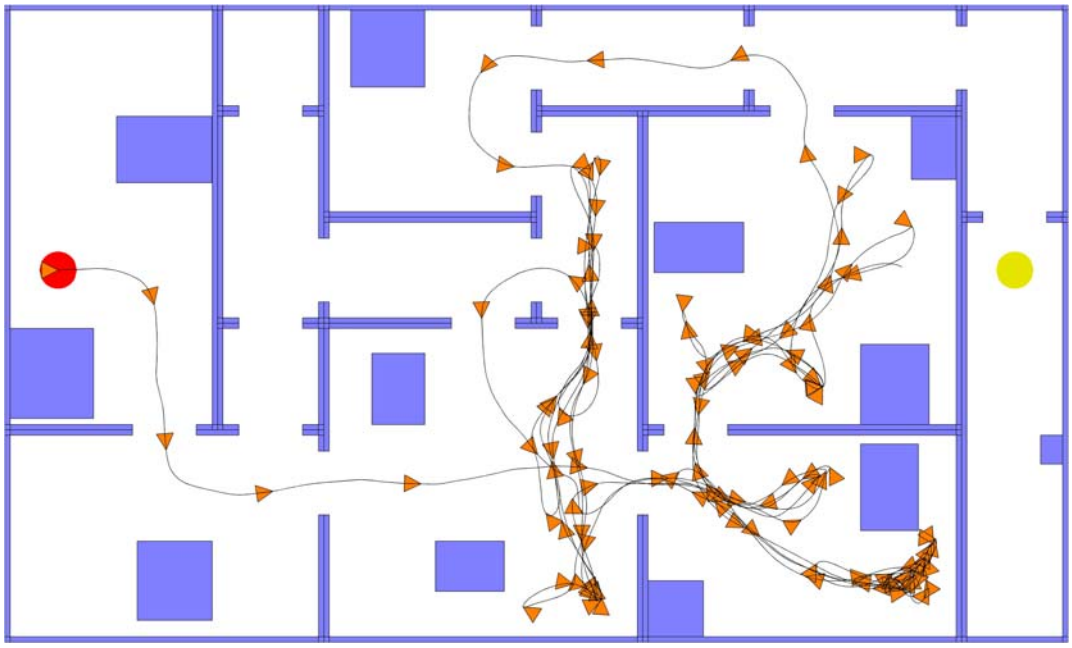

Figure 9.59: Path travelled when the safety weight = 1.

9.3.3 Confusion and Introspection

Confusion and introspection are tested in environment set G, which contains undetectable objects. The positions of these obstacles are initially unknown to the robot, so the robot constructs a danger map by recording points of collision.

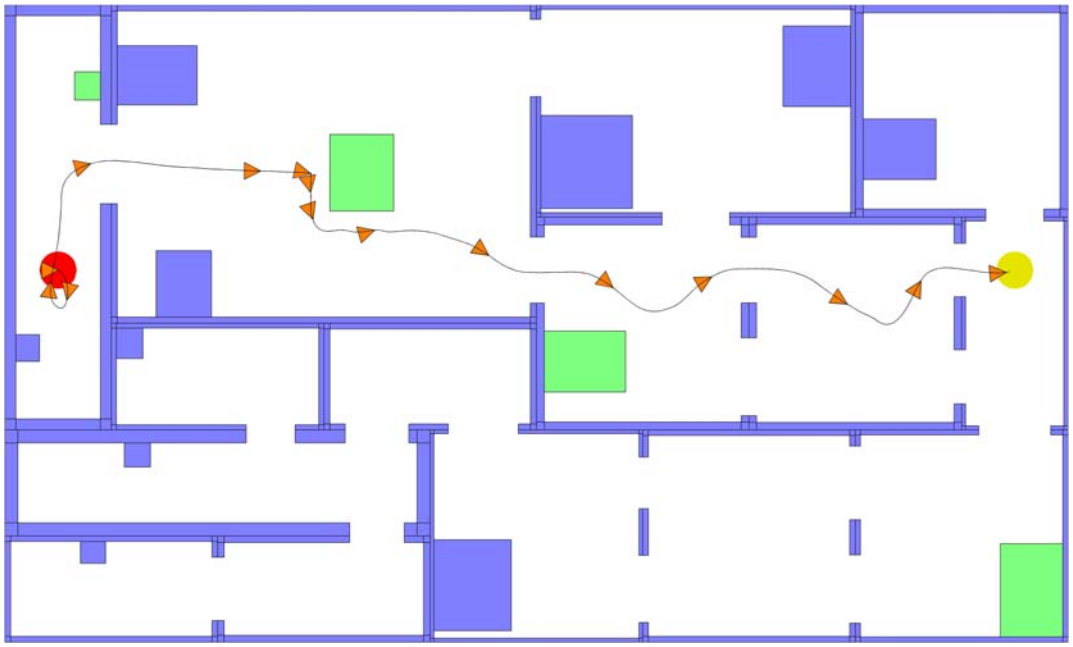

Figure 9.60: Path travelled when the introspection weight = 0.

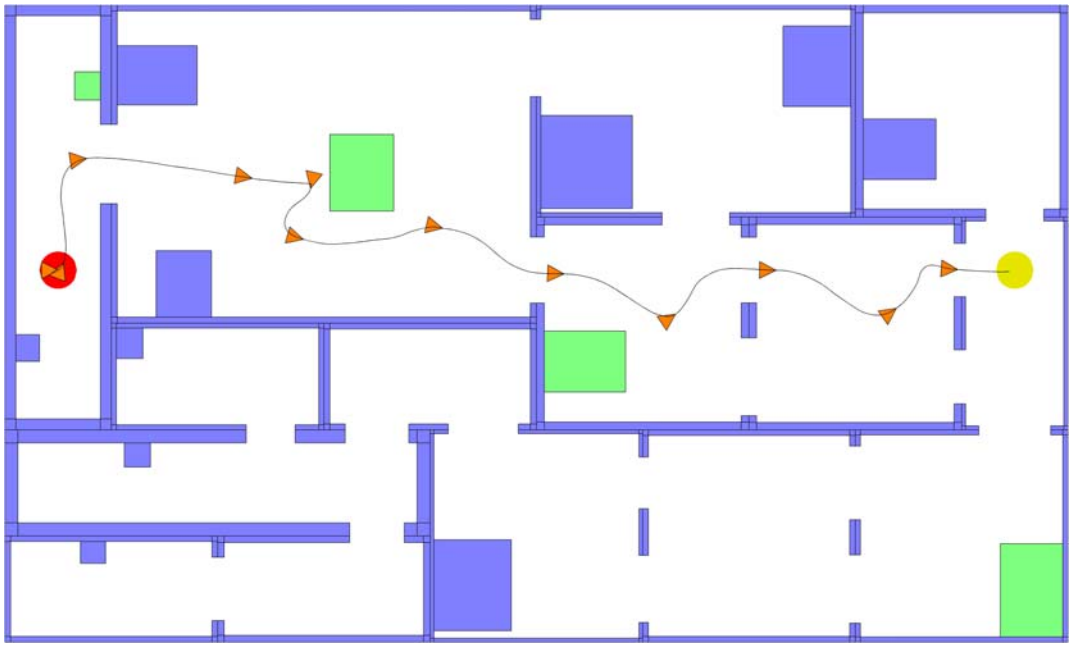

Figure 9.61: Path travelled when the introspection weight = 1.

Initially, mapped confusion is disabled (by setting a weight controlling its mapped/global bias to 1) and the contribution of the introspection drive is varied. When this contribution is low (Figure 9.60), the robot tends to collide multiple times with each unseen obstacle. Conversely, when it is high (Figure 9.61), confusion is only activated following a collision with an unseen obstacle, so it does not prevent all

collisions. Nevertheless, once a collision does occur, the robot's aversion to mapped obstacles and points of collision increases, encouraging it to give them a wide berth.

This causes a general decrease in collisions as the contribution of introspection increases (Figure 9.62). However, the minimum of around 2 collisions per minute remains unacceptably high. The improvement comes at minimal cost to completion time (Figure 9.63).

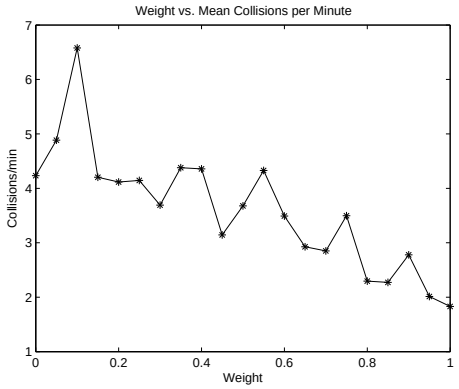

Figure 9.62: Mean collisions per minute for 20 experimental runs per introspection weight.

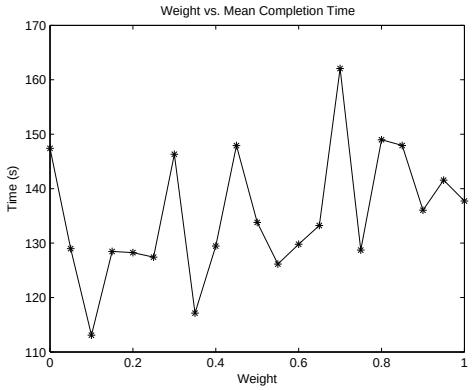

Figure 9.63: Mean completion time vs. introspection weight.

Next, introspection is maximally enabled, while a weight controlling the bias between mapped confusion and global confusion is varied from 0 (entirely mapped) to 1 (entirely global). Five passes (iterations) through the environment are executed in each experimental run, to allow the robot time to update its confusion and danger maps. When mapped confusion is disabled, the robot tends to collide with each undetectable obstacle in its path at least once. In the example shown in Figure 9.64 global confusion induces cyclical motion, causing multiple collisions during each iteration. When the robot collides with an obstacle, its aversion to mapped obstacles increases and it turns away. However, when confusion and introspection fade, it turns back towards the obstacle and collides with it again, restarting the process. These problems are reduced by mapped confusion, which typically prevents the robot from colliding with unseen obstacles with which it has previously collided (Figure 9.65).

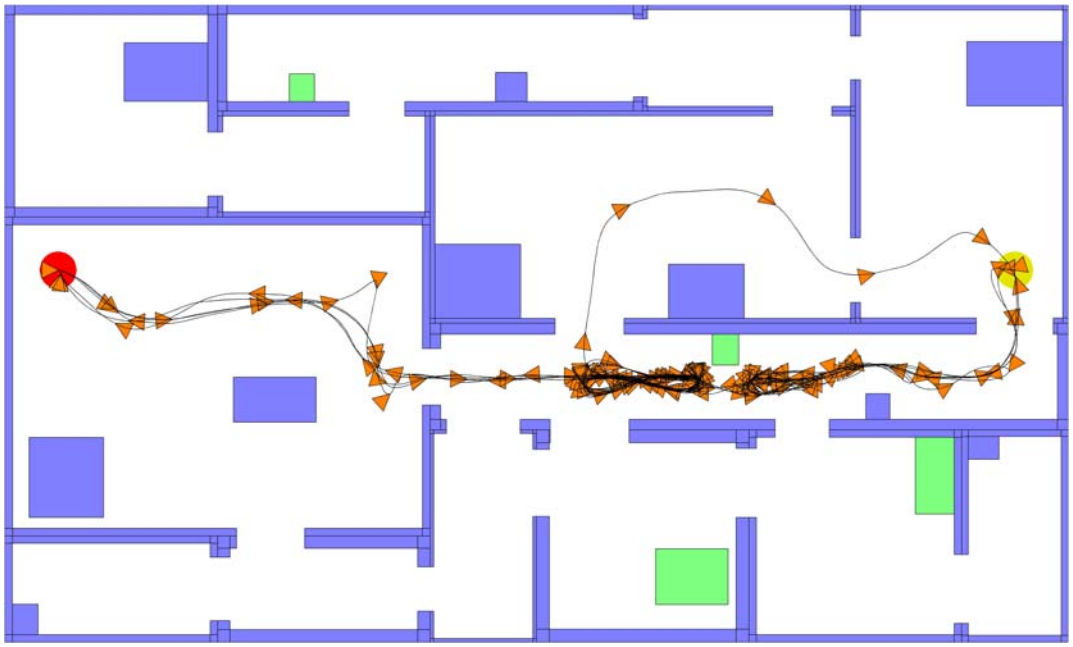

Figure 9.64: Path travelled when $\varphi = 1$.

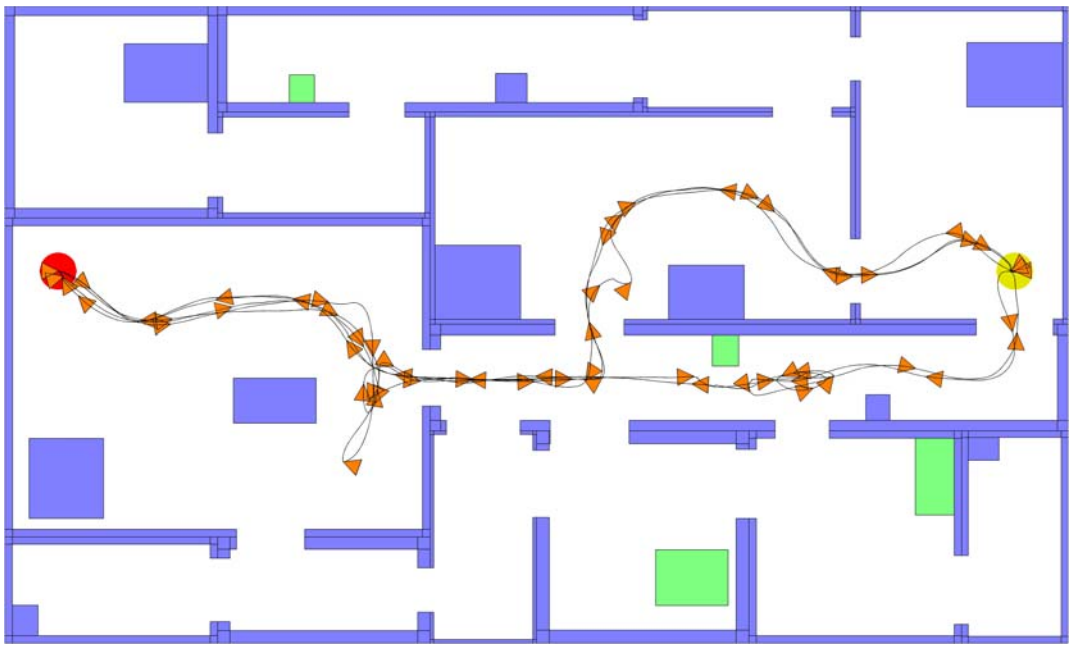

Figure 9.65: Path travelled when $\varphi = 0$.

Figures 9.66 and 9.67 show the resulting collision rate and completion time as the weight is varied. Data shown represent the average of all iterations, and of only the final two iterations. A general improvement in collision rate (at minimal cost to completion time) is apparent in both cases as the robot becomes more heavily reliant on mapped confusion, but the lowest collision rates (approximately 0.5 collisions per

minute) occur during the later iterations, once the robot has more extensively updated its danger and confusion maps. Figure 9.68 shows this general improvement as the robot learns about its environment. The improvement is more pronounced with mapped confusion (weight = 1) than global confusion (weight = 0).

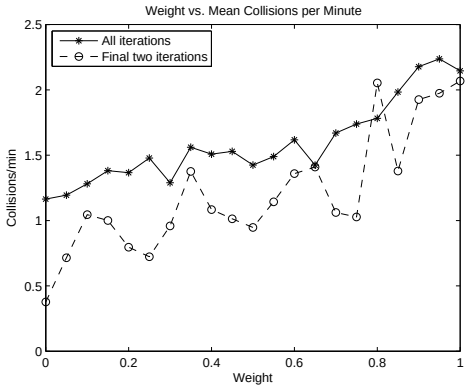

Figure 9.66: Mean collisions per minute for 20 experimental runs per φ value.

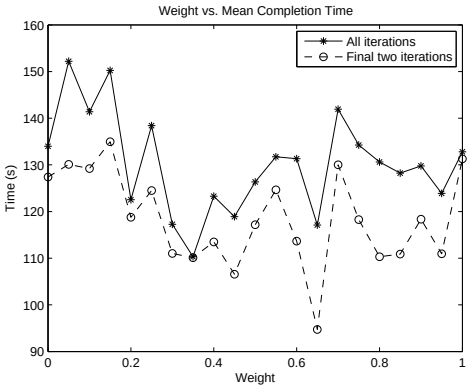

Figure 9.67: Mean completion time vs. φ.

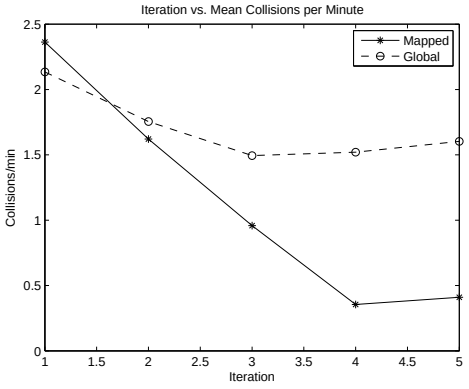

Figure 9.68: Mean collisions per minute for 20 experimental runs per iteration.

9.4 Summary

Navigation performance contributions of various affective components have been presented. The purpose of these components is to adapt the robot's parameters to respond appropriately to its environment, and to cope with certain situations it encounters without compromising its performance during normal operation. Results in this chapter show that many affective components are successful in that regard, yielding performance that compares favourably with navigation using constant parameters.

The next chapter includes an analysis of the robot's behaviour when these components are enabled simultaneously. Some extensions to the model and its application will also be investigated.

10 Integrated Navigation Experiments

Chapter 9 demonstrated the contributions of various individual affective components to navigation performance, but it did not address the behaviour of the fully integrated model. In most complex systems the behaviour of a component functioning in isolation does not necessarily reflect its behaviour when operating in unison with other components. This is also true for the affect model. Some components may interact in an unpredictable manner, while others are expected to perform functions that can overlap (e.g. curiosity and mapped emotions). It is worthwhile to assess the system's performance with all of these components simultaneously enabled so that any emergent behaviours or internal conflicts can be investigated.

10.1 Demonstration

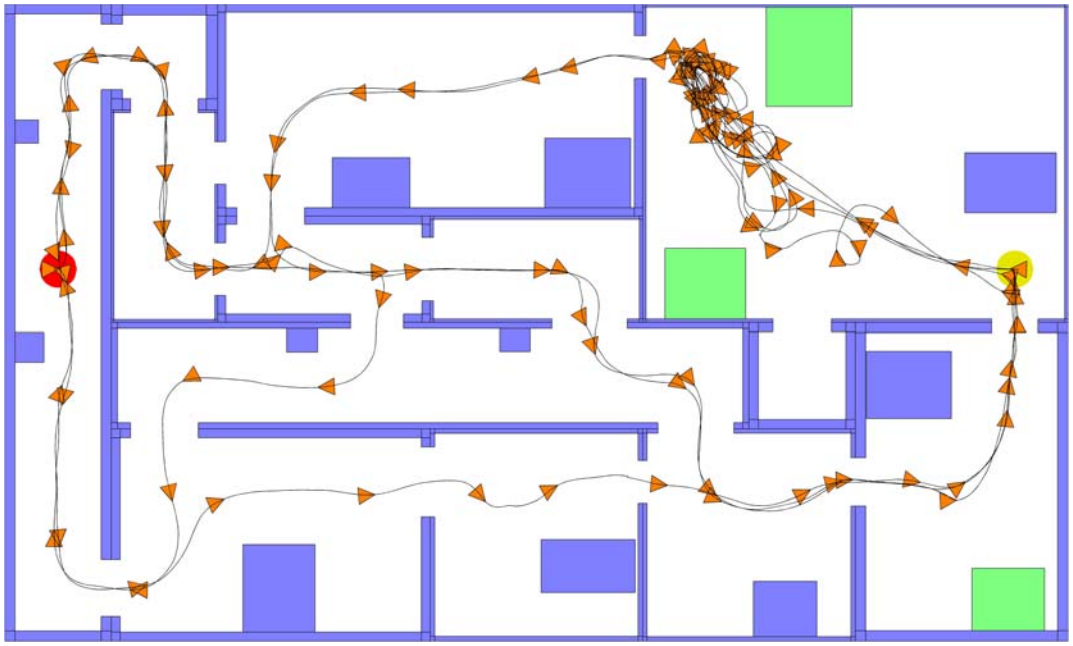

Figure 10.1: Path travelled through an environment in set G.

Before we begin a performance analysis, the interactions between the different affective components are demonstrated during five traversals of one of the environments in set G (which contains undetectable obstacles and random width doorways and walls). This environment is selected as a representative example because it elicits the full range of affective responses and presents a significant

navigational challenge. All stimuli, emotions, moods and drives are enabled in this experiment. The resulting path travelled by the robot is shown in Figure 10.1.

During the robot's journey, the intensities of all of its stimuli, emotions, moods and drives are recorded. Figure 10.2 shows the resulting emotion intensities and each of their eliciting stimuli. These intensities are not only a function of the affective stimuli, but also of previous emotions elicited in the robot's current location. Thus, while the stimuli shown in Figure 10.2 always elicit emotional responses, subsequent emotional responses can occur in their absence.

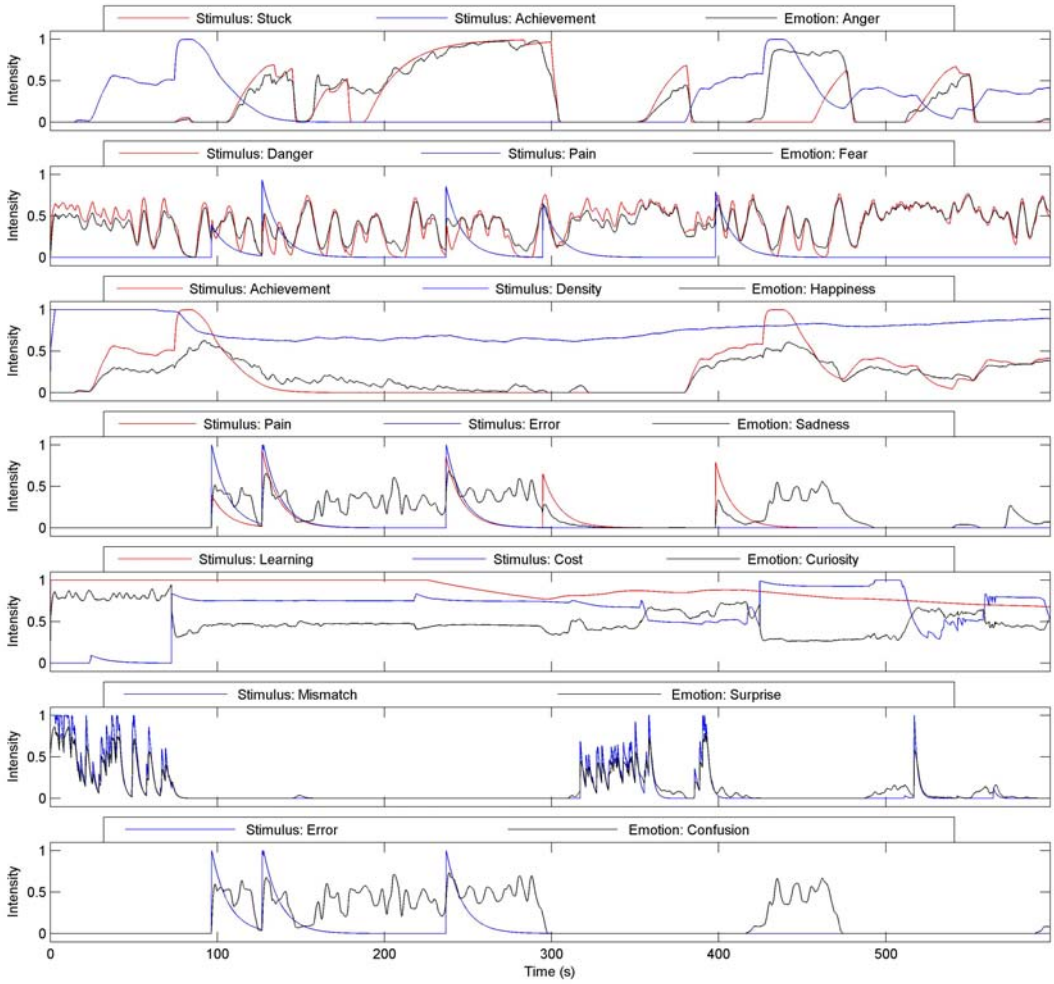

Figure 10.2: Interactions between stimuli and emotions for an environment in set G.

Anger is increased by the stuck stimulus and reduced by achievement. Whenever the robot becomes obstructed (e.g. by a narrow doorway), the stuck stimulus begins to increase, which causes growing anger. Both the stimulus and emotion quickly subside once the robot escapes the obstruction. In this example, brief obstructions occur on a

number of occasions, but one obstructed state persists for an extended period of time when the robot is unable to turn into the doorway in the northeast corner of the map. This obstruction is exacerbated by the nearby undetectable obstacles, as the robot increases its map-based avoidance weights in their presence to prevent further collisions with them. Anger arises at several points despite a lack of a stuck stimulus, when the robot is near locations where anger was previously elicited and applied to the anger map. The robot's progress is generally not sufficient to consistently produce high achievement in this environment. Hence, its contribution is minimal.

Fear is elicited by danger (a function of local obstacle proximities) and pain (a function of collisions). The environment is very confining, so the danger stimulus is frequently highly activated throughout the journey. Collisions (and therefore pain) begin to occur when the robot encounters the undetectable obstacles and certain narrow doorways.

Happiness is increased by achievement and decreased by density, which is an estimate of the average occupancy probability of the entire explored space. Since the environment is relatively cluttered, density remains consistently high. Combined with the rarely-elicited achievement stimulus, this generally produces low levels of happiness.

Sadness is a function of the pain and error stimuli. In this example, the robot collides with unseen obstacles three times, eliciting both pain and error. The other two collisions are with doorway corners and thus elicit only pain.

Curiosity is a function of the learning stimulus. Learning is initially high due to the high rate of knowledge acquisition during exploration, but it slowly decreases once the majority of useful information has been acquired. Surprise visibly decreases when the cost stimulus increases, so that high exploration tendencies are only allowed when they do not significantly degrade the quality of paths planned.

Surprise is primarily dependent on the mismatch stimulus. The environment is initially unexplored, so mismatch is frequently high during the first traversal through the environment, and again when it travels rapidly through other unexplored areas. Surprise is generally not highly elicited when the robot lingers in an area for a sufficient duration to allow its map to align with local sensor data.

Confusion increases largely due to the error stimulus. Once the robot begins to collide with undetectable obstacles, confusion starts to have a significant effect over the robot's behavior, causing its obstacle avoidance and planning systems to rely more heavily on internal representations that incorporate mapped points of collision. Although it cannot prevent initial collisions with obstacles that the robot cannot detect, confusion dissuades the robot from repeating them.

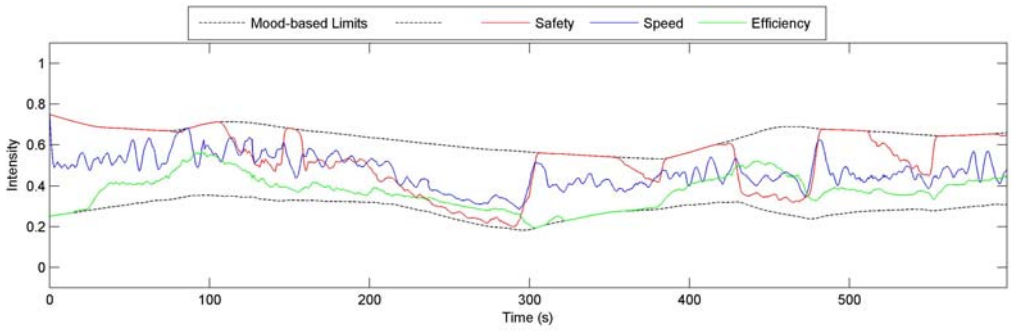

Figure 10.3: Survival drives bounded by positive and negative moods for an environment in set G.

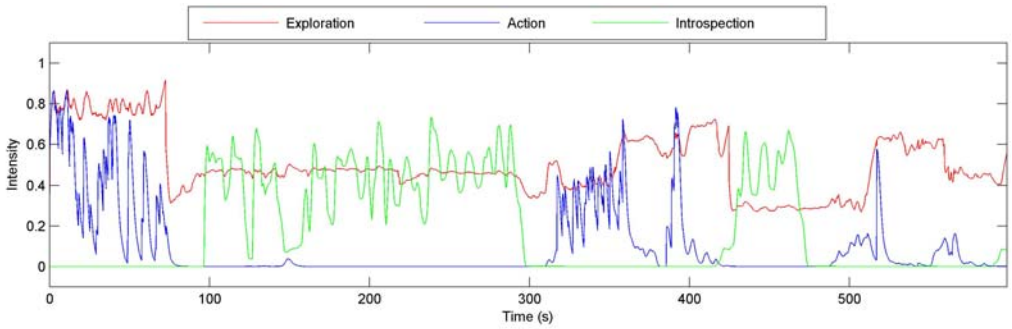

Figure 10.4: Unbounded strategic drives for an environment in set G.

Negative mood is a function of anger, fear and sadness, while positive mood is only dependent on happiness. The three survival drives (safety, speed and efficiency) constrained by these moods are shown in Figure 10.3, along with their mood-based limits. Moods grow and decay over a significantly longer timescale than their eliciting emotions. These moods are initialised to 0.5, so the drives initially vary within the range 0.25 – 0.75. Positive mood gradually declines due to the robot's slow progress, until it begins to converge on the goal more rapidly in later iterations. Negative mood tends to be dominated by anger, which is the only contributing emotion that tends to persist at very high levels for extended periods of time. Hence, negative mood

increases when the robot becomes severely obstructed, slowly decreasing the limit constraining its safety drive until it can escape the obstruction.

The three strategic drives (shown in Figure 10.4) are not constrained by moods, so they can always span the full interval [0, 1]. These interactions can be approximated as direct mappings between emotion-drive pairs (curiosity-exploration, surprise-action and confusion-introspection), even though exploration and action receive inputs from multiple emotions.

A second demonstration is conducted in one of the environments in set A (Figure 10.5), which represents a very unchallenging navigation scenario. The resulting stimuli, emotions, moods and drives are shown in Figures 10.6-10.8.

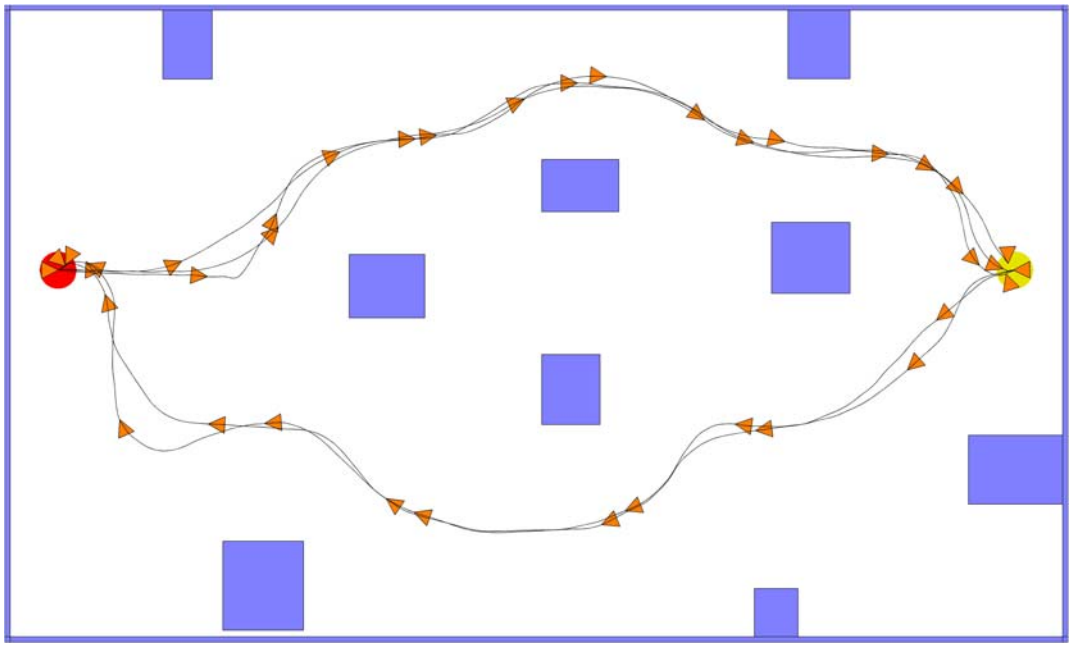

Figure 10.5: Path travelled through an environment in set A.

In this environment, there are no narrow openings or narrow maze-like paths to obstruct the robot's progress, and few obstacles to pose a danger to high-speed travel. Hence, the achievement stimulus remains high throughout each journey, while stuck, danger, pain, error and density are typically low, or zero (Figure 10.6). As a result, happiness tends towards its maximum value, while anger, fear and sadness tend towards low or minimum values. This results in a gradual increase of positive mood, and a decrease of negative mood (Figure 10.7). Once the robot has traversed the environment five times, they have almost reached their maximum and minimum points, respectively. Among the remaining stimuli and emotions, learning and

curiosity are the only ones that remain high for a significant duration. The five traversals are completed too quickly for the learning stimulus to subside, so the robot's exploration drive remains high throughout the demonstration (Figure 10.8). This is somewhat countered by the mapped emotions' strong bias towards paths that have previously elicited happiness, so the robot tends to adhere to the two 'primary' paths between the goal points.

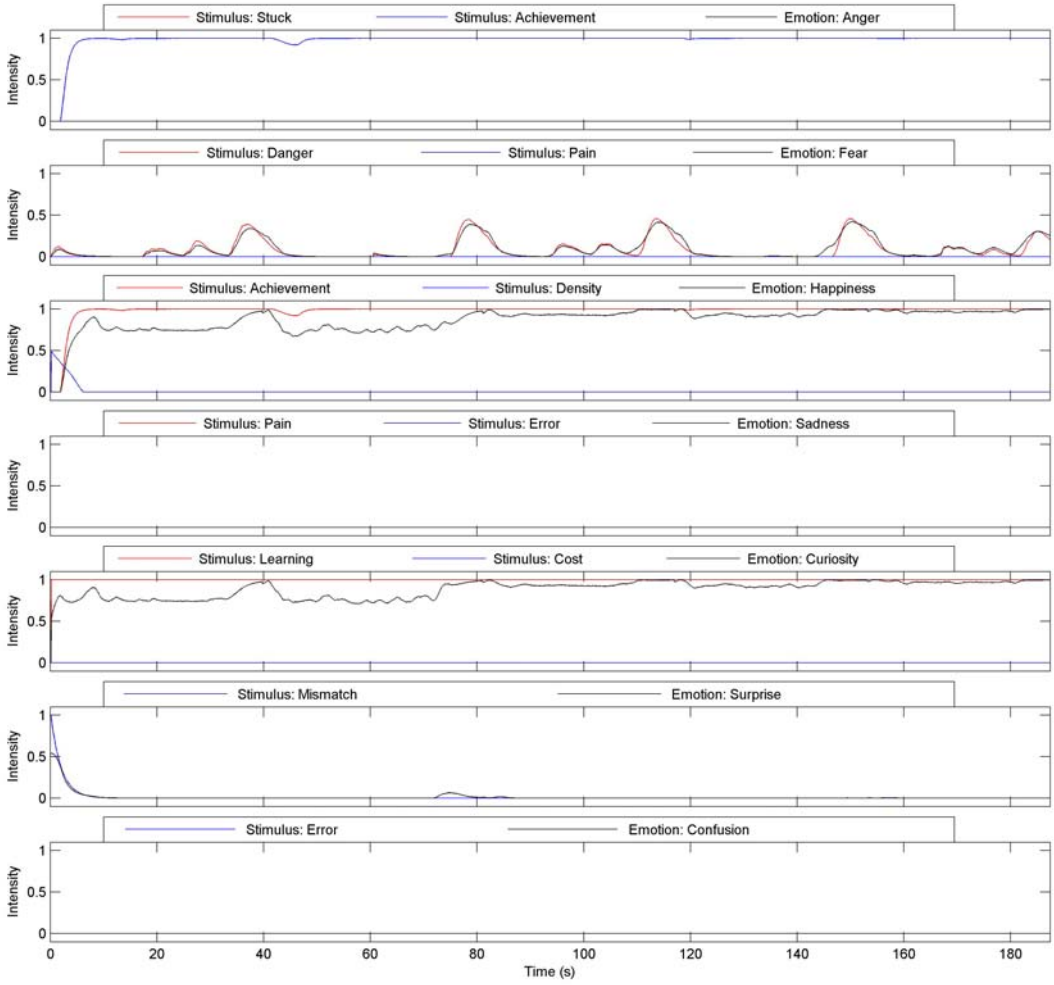

Figure 10.6: Interactions between stimuli and emotions for an environment in set A.

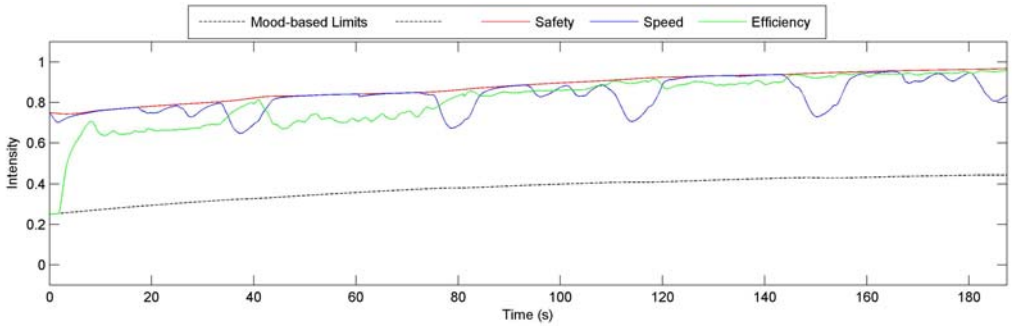

Figure 10.7: Survival drives bounded by positive and negative moods for an environment in set A.

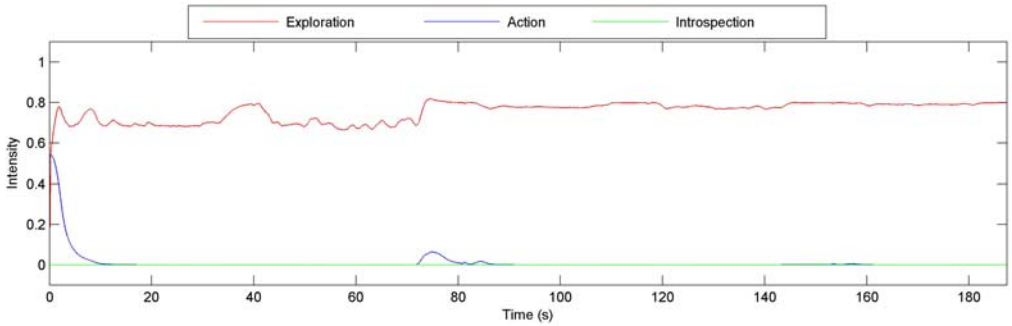

Figure 10.8: Unbounded strategic drives for an environment in set A.

10.2 Performance Analysis

A quantitative analysis of the integrated affect model's contributions to adaptive performance is conducted by comparing the robot's behaviour in two different configurations. In the first configuration, the drive parameters are maintained at constant values, representing a robot without affective influences. Survival drives are set to their equilibrium position of 0.5, while strategic drives are set to 0. All emotions and moods are set to 0. In the second configuration, affective parameter modulation is fully enabled as demonstrated in the previous section, affording the robot its full range of emotional responses.

To thoroughly assess performance, these experiments are repeated in a diverse selection of environment sets. Five experimental runs are performed for each of the 20 environments in a given set, producing 100 samples per configuration/environment set combination. The robot is instructed to travel back and forth between two points, for a total of nine traversals (iterations) per experimental run. At the beginning of each run,

the robot's map and other parameters are reset to their initial conditions, so it must explore and adapt its parameters while it traverses the environment.

10.2.1 Environment Set A

In environment set A (example paths are shown in Figures 10.9 and 10.10), the configuration with parameters modulated by the affect model outperforms the one with constant parameters in every respect (except collisions, as none occur in both cases). As shown in Section 10.1, the robot generally elicits high intensities of happiness and low levels of fear, anger and sadness. Thus, its speed limit is relaxed, and it reduces the amount of data processed each control cycle, yielding lower completion times (Figure 10.11), higher average velocities (Figure 10.12) and lower execution time ratios (Figure 10.13). The robot's incentive to explore the environment (due to curiosity) tends to outweigh its incentive to adhere to successful paths (due to mapped happiness), resulting in higher exploration coverage (Figure 10.14). The effects of an increased exploration drive are most apparent in the second iteration, where the affective robot is sometimes slower than the non-affective robot to converge on the goal (Figure 10.11).

Moods and mapped emotions take some time to converge on their final values, so initially the two systems are more similar in behaviour (aside from the affective system's explorative tendencies). As time progresses, the systems become more divergent, and the affective system becomes more clearly advantageous.

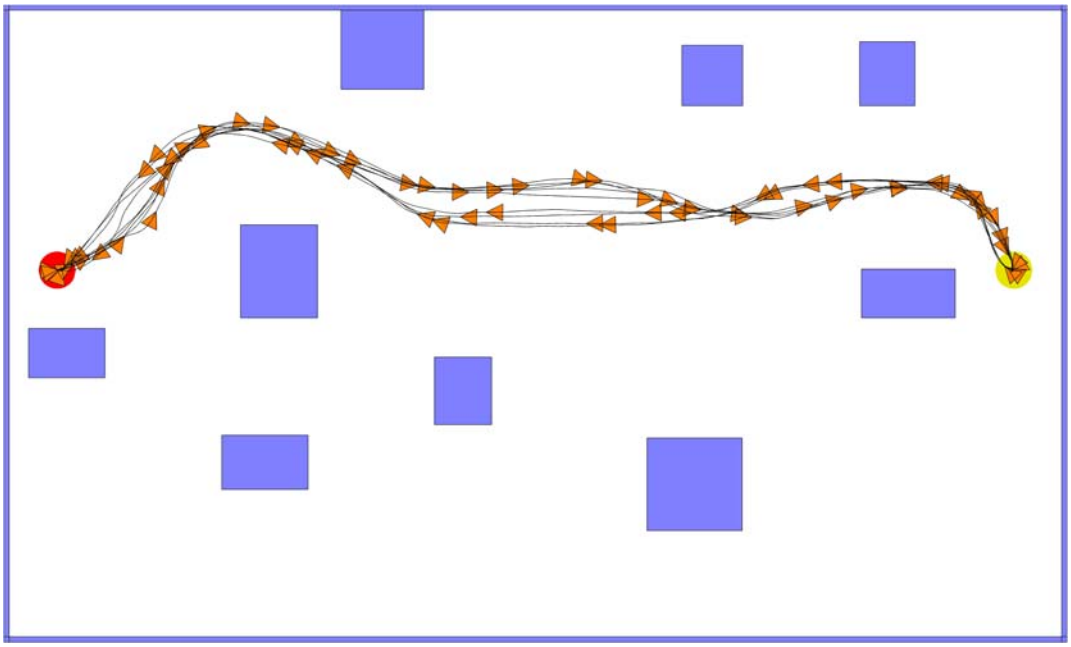

Figure 10.9: Set A – Example path resulting from constant parameters.

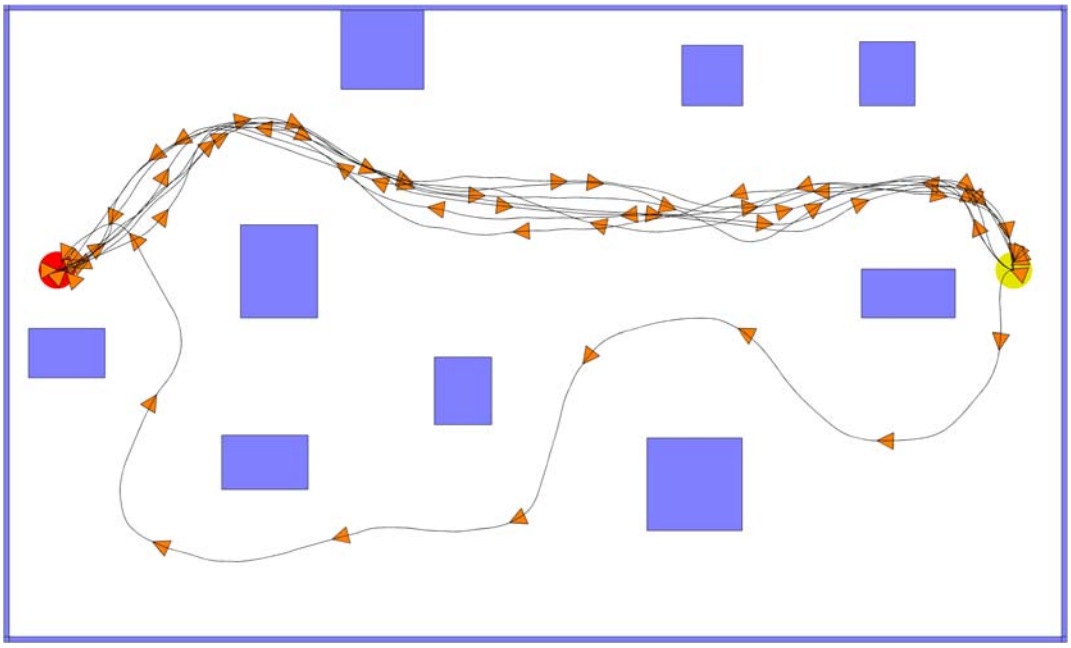

Figure 10.10: Set A – Example path resulting from affective parameter modulation.

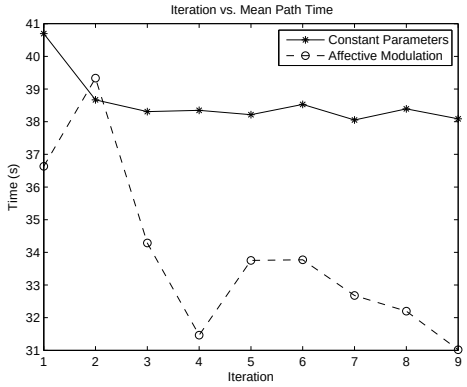

Figure 10.11: Set A – Mean completion time for 100 experimental runs per iteration.

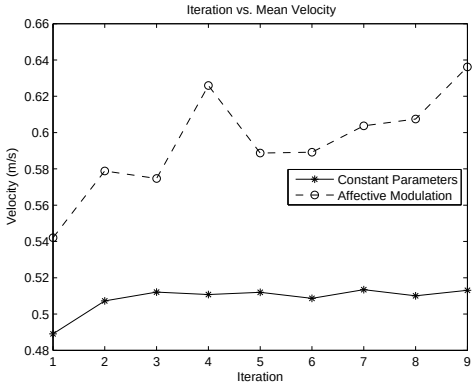

Figure 10.12: Set A – Velocity.

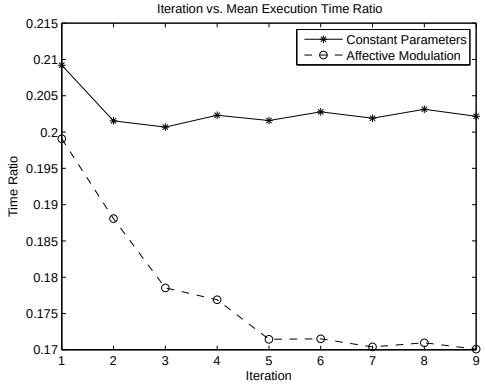

Figure 10.13: Set A – Execution time ratio.

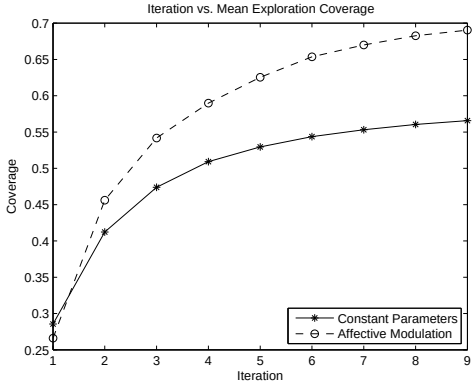

Figure 10.14: Set A – Exploration coverage.

10.2.2 Environment Set B

Operating in set B (Figures 10.15 and 10.16), the system with constant parameters initially outperforms one with affect-modulated parameters. However, once its moods and mapped emotions have stabilised and the area has been explored, the affective system gains the performance advantage. These environments present a moderate navigational challenge, so happiness tends to be lower than in sparsely occupied environments, while fear and anger are sometimes higher (sadness remains low, due to the absence of collisions).

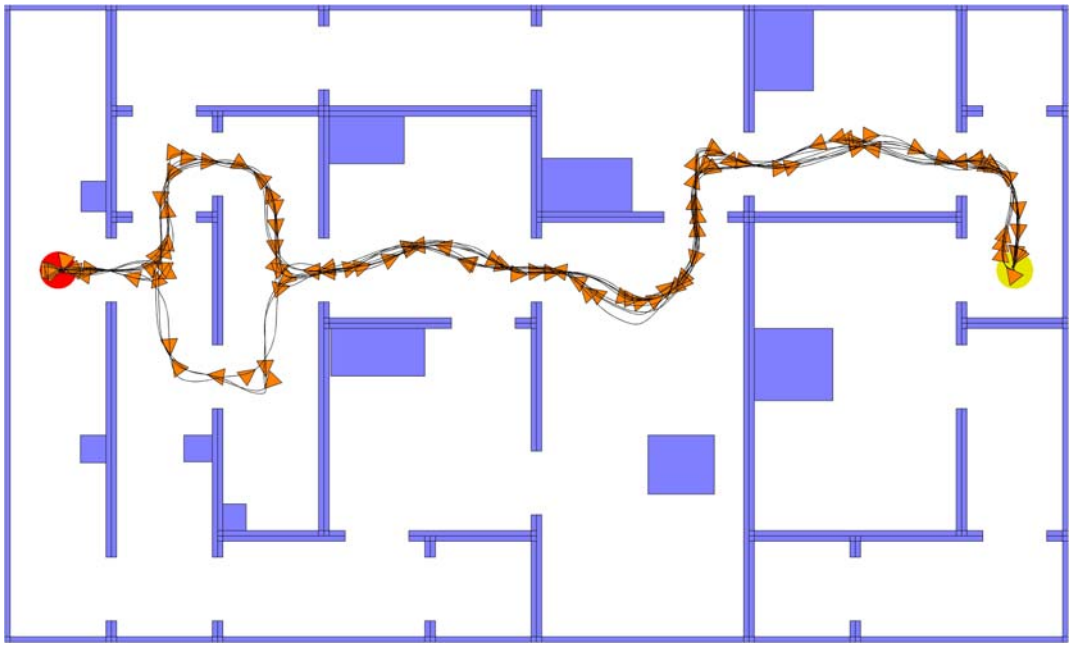

Figure 10.15: Set B – Example path resulting from constant parameters.

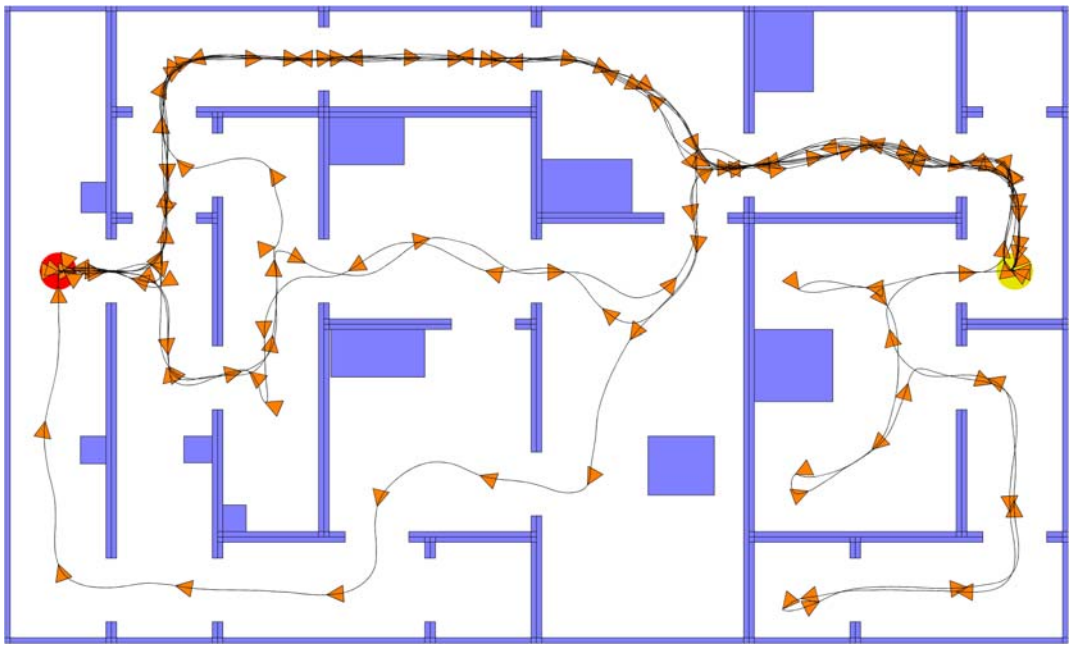

Figure 10.16: Set B – Example path resulting from affective parameter modulation.

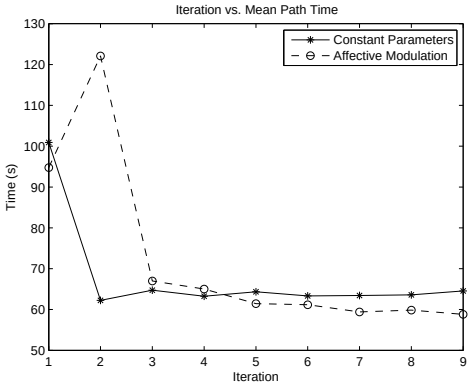

Figure 10.17: Set B – Mean completion time for 100 experimental runs per iteration.

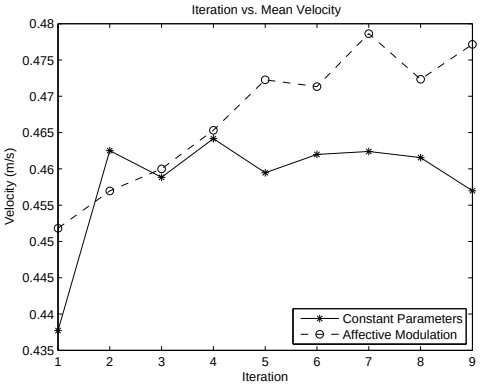

Figure 10.18: Set B – Velocity.

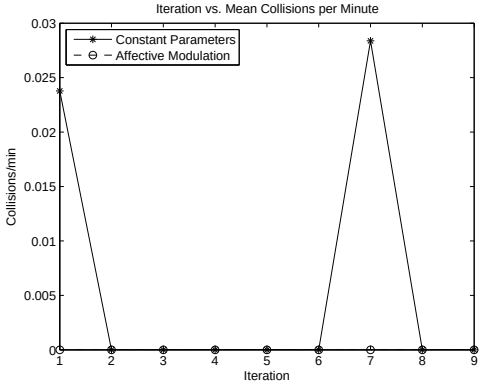

Figure 10.19: Set B – Collisions per minute.

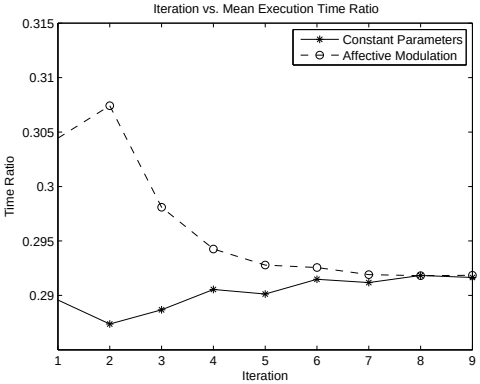

Figure 10.20: Set B – Execution time ratio.

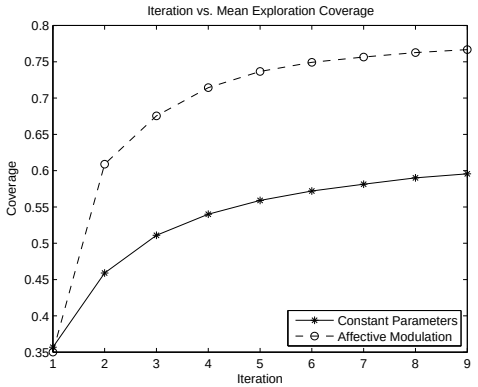

Figure 10.21: Set B – Exploration coverage.

Initial high levels of curiosity result in a significant time investment (Figure 10.17) during early iterations (particularly the second one). The affective system tends to settle on slightly higher velocities than the static one (Figure 10.18). Combined with the additional world knowledge gained through increased exploration (Figure 10.21),

this yields improved completion times during later iterations (Figure 10.17). Two collisions occurred during these experiments under the constant parameter configuration, while the affective configuration resulted in none (Figure 10.19). The nominally higher safety drive produced by the affective system may have contributed to this improvement. Affective parameter modulation initially results in higher execution time ratio than constant parameters (Figure 10.20), due to initial slow progress during exploration of a new environment. As the robot's progress improves, the exploration time ratio converges to around the same level as the static system. This is loosely analogous to the tendency for humans to apply greater effort and concentration to new tasks, while devoting less attention to those tasks once they have been repeated multiple times and formed more efficient neural connections.

10.2.3 Environment Set C

Set C (Figures 10.22 and 10.23) reveals some potential problems with the integrated affective system. While the experiments with survival drives and their associated emotions shown in Section 9.2 indicated that they have a slight positive overall effect on performance in these environments, the addition of other emotional influences appears to have cancelled the improvements out. In particular, gains to completion time in early iterations are likely nullified by the combination of emotion maps and a high exploration drive (Figure 10.24). The affective system also generally favours slightly lower velocities than the static system (Figure 10.25). Both completion times and velocities appear to converge in the final iterations, and thus the robot's performance becomes similar. Extending the duration of the experiments from five iterations (in Section 9.2) to nine has also revealed a tendency for increased collision likelihood in later iterations (Figure 10.26).

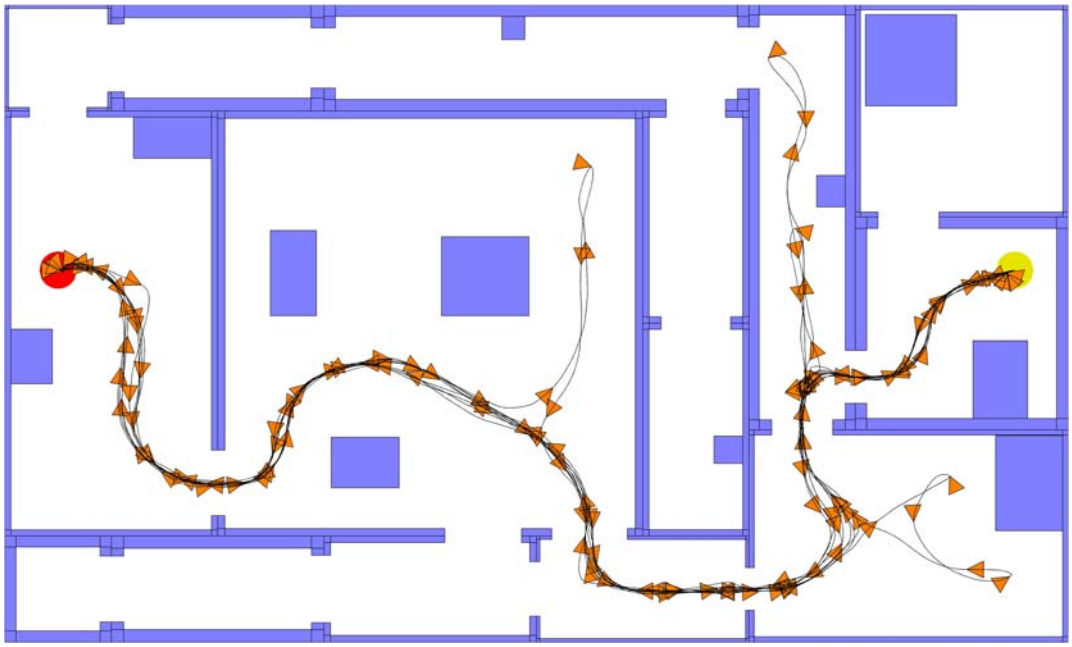

Figure 10.22: Set C – Example path resulting from constant parameters.

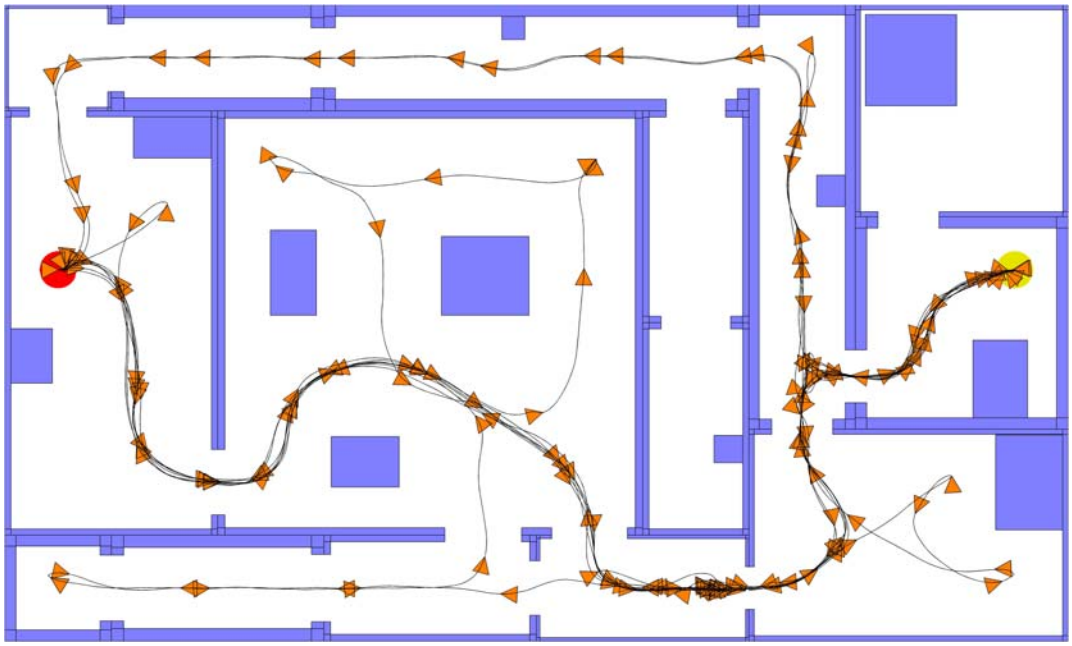

Figure 10.23: Set C – Example path resulting from affective parameter modulation.

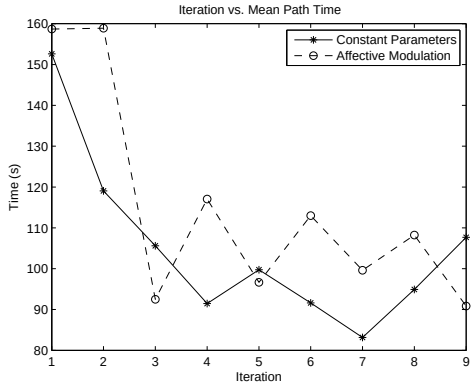

Figure 10.24: Set C – Mean completion time for 100 experimental runs per iteration.

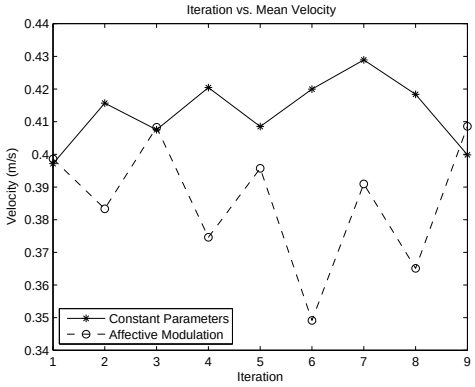

Figure 10.25: Set C – Velocity.

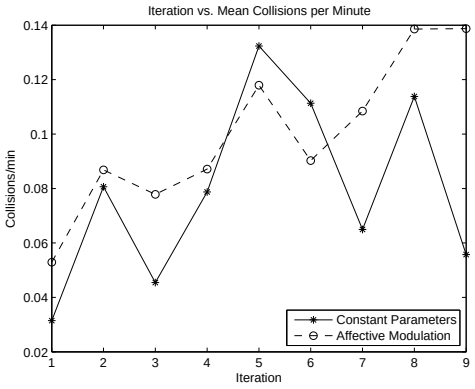

Figure 10.26: Set C – Collisions per minute.

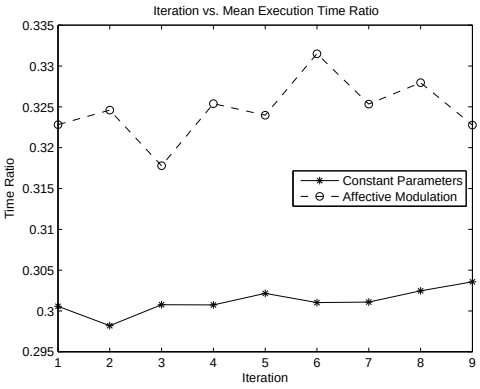

Figure 10.27: Set C – Execution time ratio.

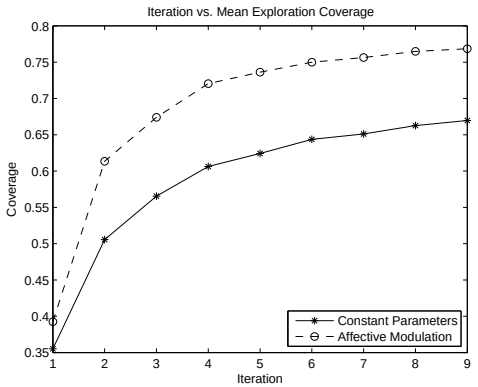

Figure 10.28: Set C – Exploration coverage.

By the final iterations, mapped anger is often high throughout large portions of each environment, which may be causing the robot to ‘overreact’ by excessively lowering its safety drive. This could be improved by increasing the decay rate of mapped anger, but it may come at a cost of increased obstructions. Execution time ratios (Figure

10.27) are consistently higher for the affective system, as the navigational challenge of these environments is typically high, and thus goal convergence is comparatively slow. One performance characteristic that is consistently improved by the affect model is exploration coverage (Figure 10.28).

10.2.4 Environment Set D

The simulated humans in set D (Figures 10.29 and 10.30) can sometimes obstruct doorways, forcing the robot to either wait for them to move out of the way or find an alternative path. Due to the presence of exploration and emotion maps, the affective system is more likely to seek alternative paths than the static system. Another obvious effect of the simulated humans is the increased number of collisions compared to environment set B (which has similar characteristics but lacks the simulated humans). After the initial exploratory period, the affective system generally achieves slightly lower completion times than the unmodulated system (Figure 10.31). The difference between the two systems is smaller than that obtained in set B primarily due to more conservative velocities being selected by the affect model in response to increased fear and anger in the presence of environmental dynamics (Figure 10.32). Collisions occur at similar rates for the two systems (Figure 10.33); a minor improvement in the total number of collisions produced by the affect model is largely obscured by random fluctuations. Execution time ratios (Figure 10.34) and exploration coverage (Figure 10.35) are higher in the configuration modulated by affect.

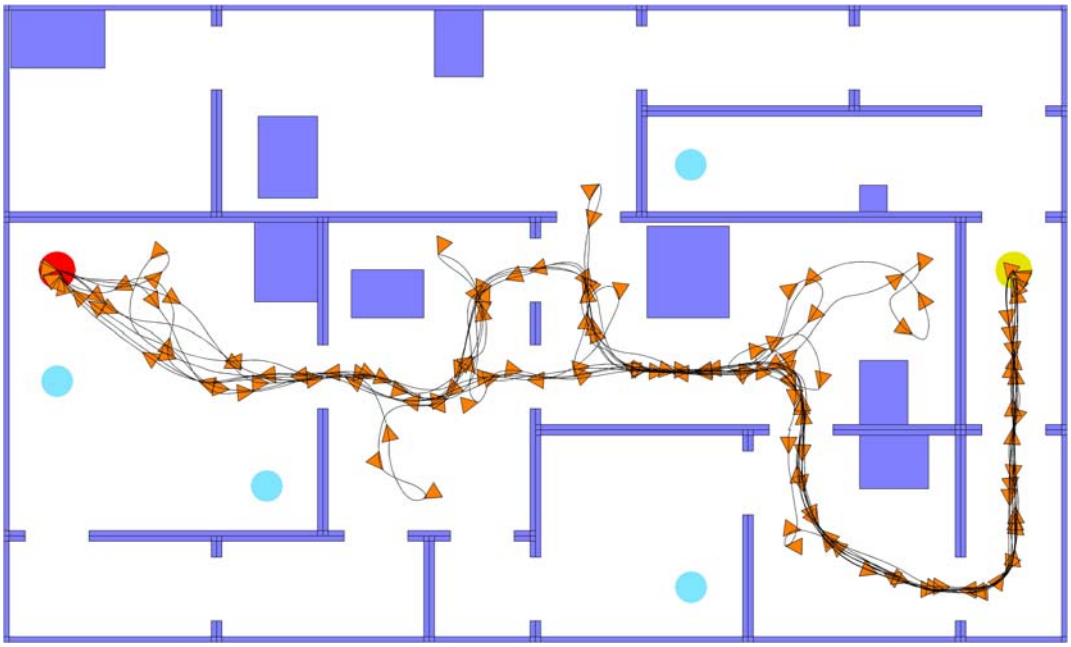

Figure 10.29: Set D – Example path resulting from constant parameters.

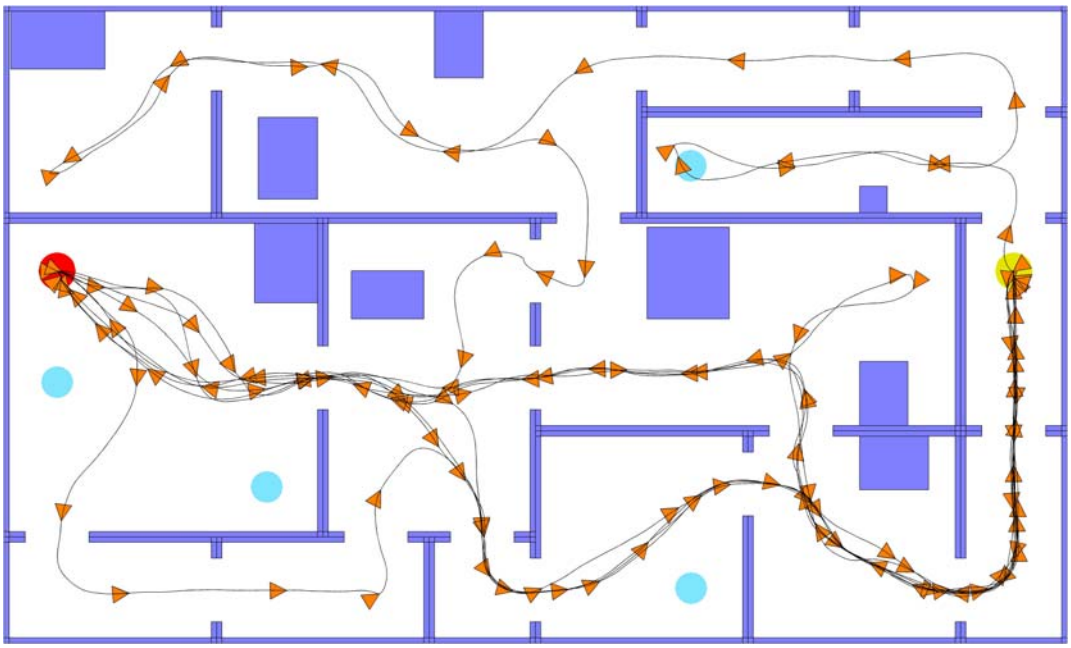

Figure 10.30: Set D – Example path resulting from affective parameter modulation.

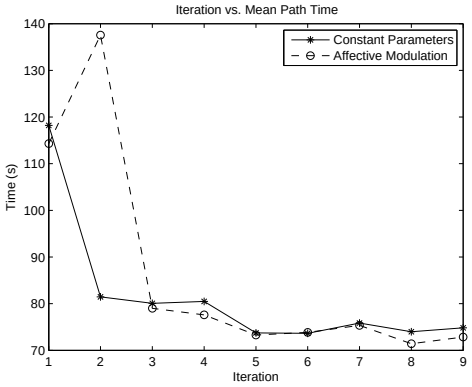

Figure 10.31: Set D – Mean completion time for 100 experimental runs per iteration.

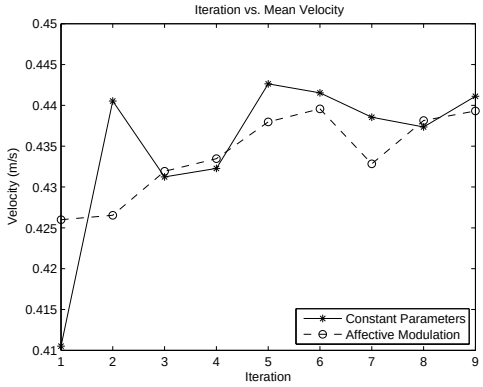

Figure 10.32: Set D – Velocity.

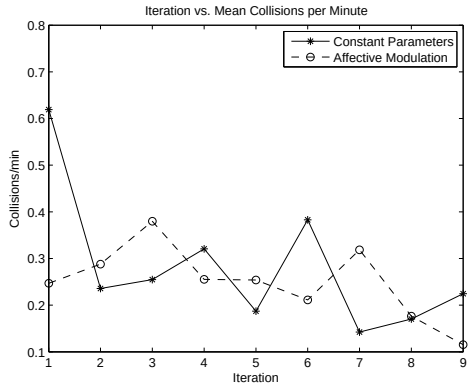

Figure 10.33: Set D – Collisions per minute.

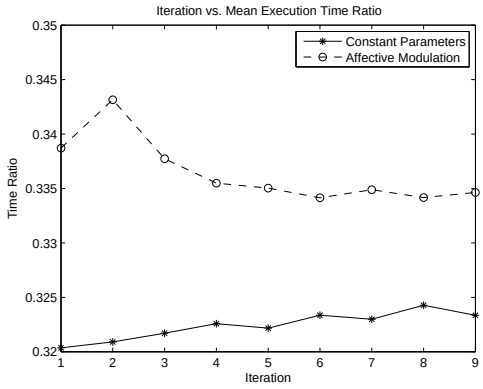

Figure 10.34: Set D – Execution time ratio.

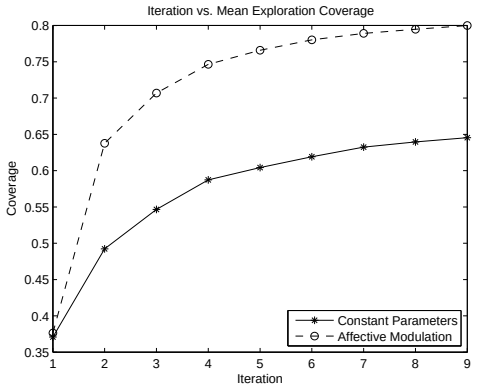

Figure 10.35: Set D – Exploration coverage.

10.2.5 Environment Set E

In set E (Figures 10.36 and 10.37), the simulated humans can actually be beneficial to some aspects of performance. While simulated humans sometimes block the robot's

path, in the same manner as they also do in set D, they can also ‘push’ the robot through narrow doorways that it might otherwise have difficulty traversing.

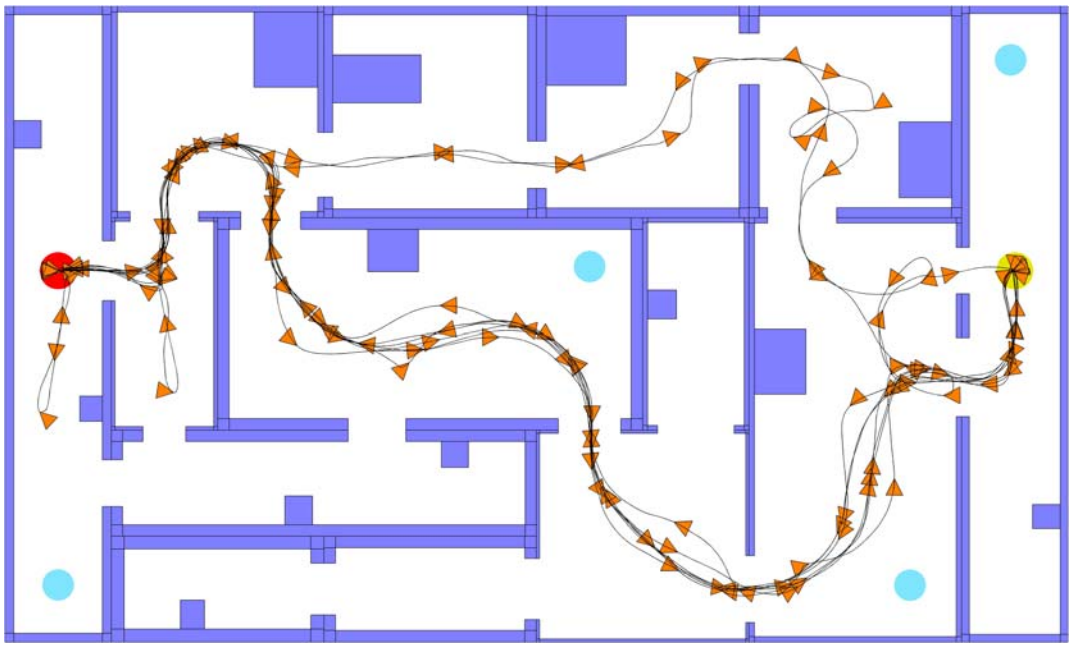

Figure 10.36: Set E – Example path resulting from constant parameters.

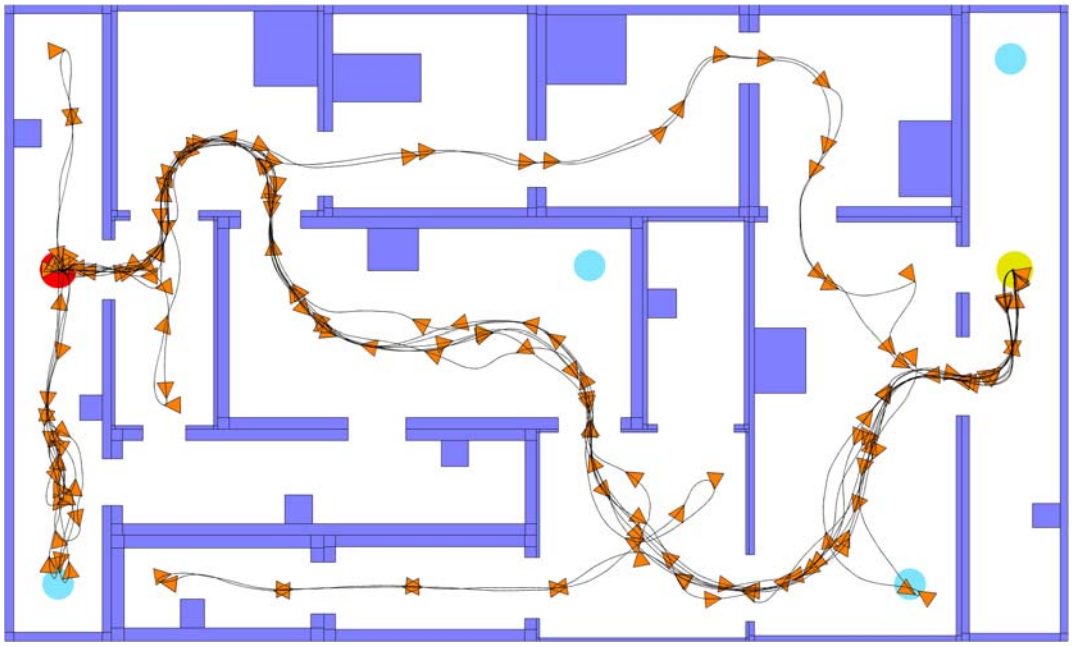

Figure 10.37: Set E – Example path resulting from affective parameter modulation.

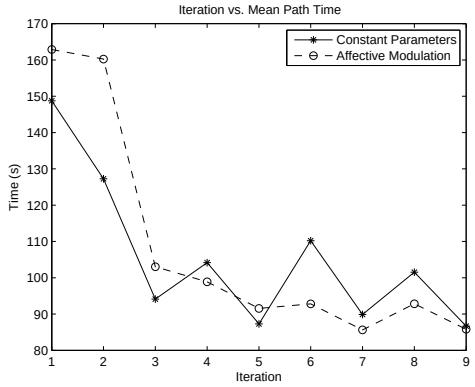

Figure 10.38: Set E – Mean completion time for 100 experimental runs per iteration.

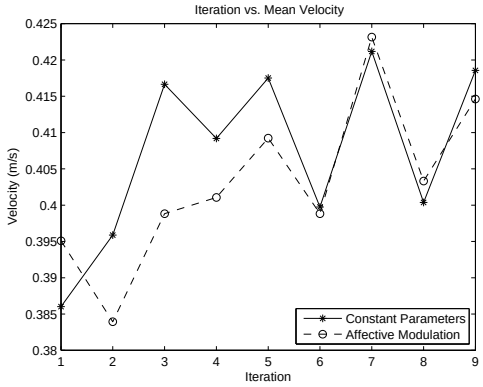

Figure 10.39: Set E – Velocity.

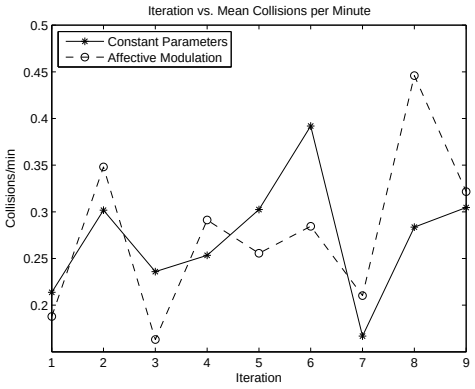

Figure 10.40: Set E – Collisions per minute.

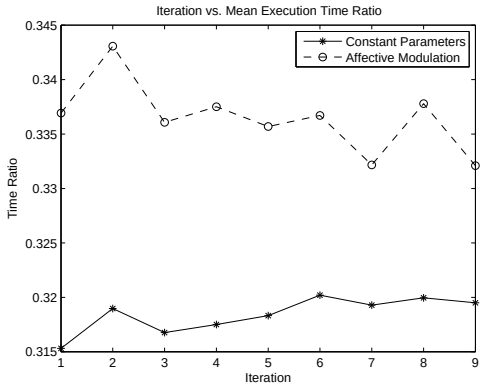

Figure 10.41: Set E – Execution time ratio.

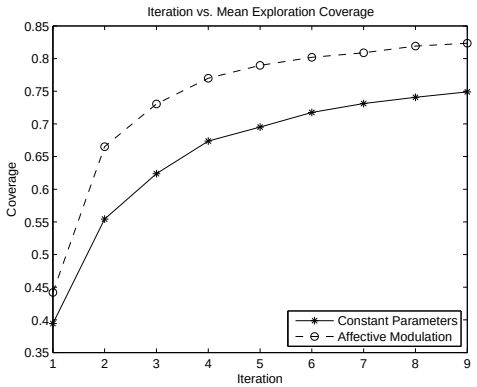

Figure 10.42: Set E – Exploration coverage.

In terms of performance, these two effects tend to largely cancel each other out, resulting in completion times (Figure 10.38) that are similar to those obtained in set C (which lacks the simulated humans). Following initial exploration, affective

modulation appears to achieve slightly lower completion times than constant parameters. The average velocities of the two systems converge on similar values (Figure 10.39). Again, a large number of collisions result from simulated human interference, but the affective system has little influence over this collision rate (Figure 10.40). Execution time ratios (Figure 10.41) and exploration coverage (Figure 10.42) remain higher in the affective configuration. However, even the non-modulated system produces relatively high exploration coverage, due to the combination of obstructive doorways and environmental dynamics.

10.2.6 Environment Set F

When the robot is tested in set F (Figures 10.43 and 10.44), the constant parameter configuration yields a very high collision rate. It does not utilize its map-based avoidance function, so the robot repeatedly collides with the same objects. While this function could be constantly enabled, it would degrade performance during normal operation. This problem is reduced by affective modulation, and in particular confusion/introspection, which highly activates the map-based avoidance function only in regions where the robot has previously collided with unseen obstacles. Thus, the robot starts to avoid unseen obstacles after it has collided with them.

Affective parameter modulation improves most performance characteristics in these environments once the robot has had time to explore and adapt. Completion times (Figure 10.45), velocities (Figure 10.46), execution time ratios (Figure 10.48) and exploration coverage (Figure 10.49) show similar trends to those obtained in set B, although the affective system tends to choose slightly more conservative parameter values in response to the increased number of collisions, particularly during early iterations. Collision rates are initially very high, but they subside as the robot adapts to its environment (Figure 10.47). While the static system rarely produces averages of less than 2 collisions per minute, the affective system's collision rate consistently decreases as the robot adds new points of collision to its danger map and updates its confusion map.

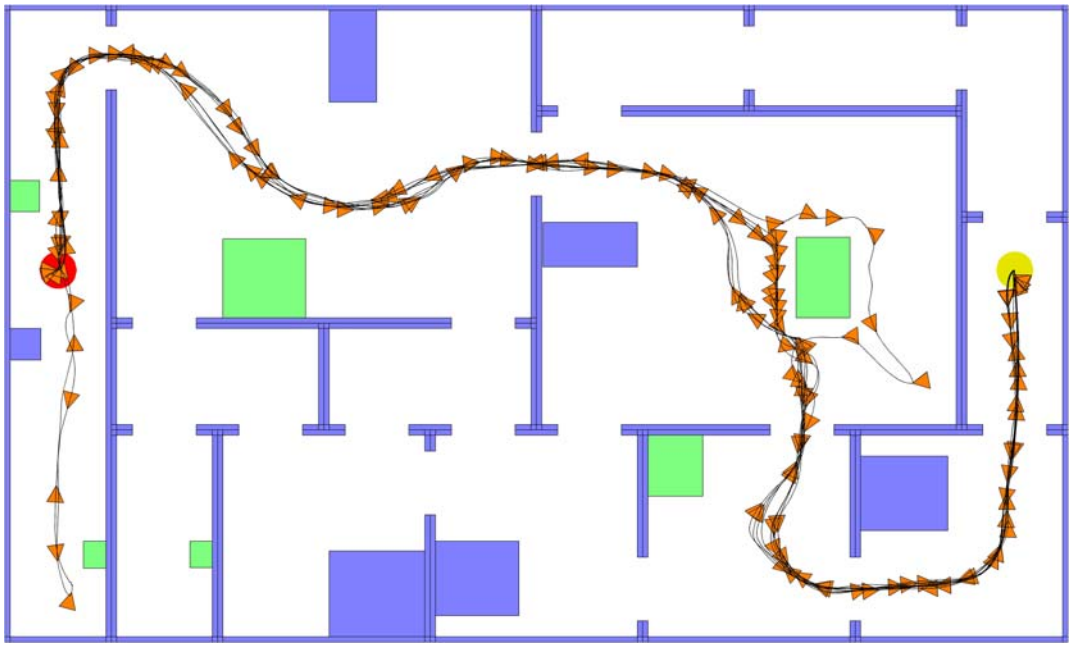

Figure 10.43: Set F – Example path resulting from constant parameters.

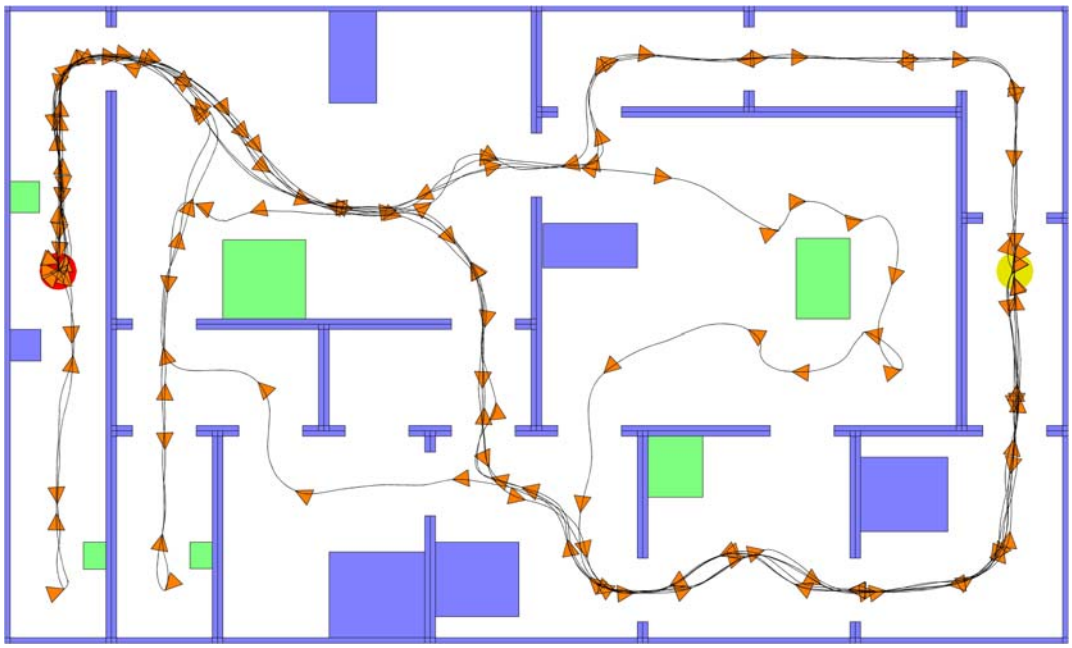

Figure 10.44: Set F – Example path resulting from affective parameter modulation.

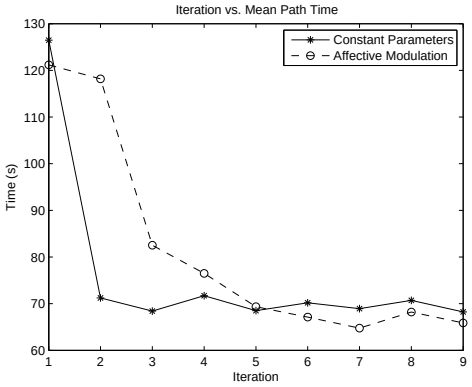

Figure 10.45: Set F – Mean completion time for 100 experimental runs per iteration.

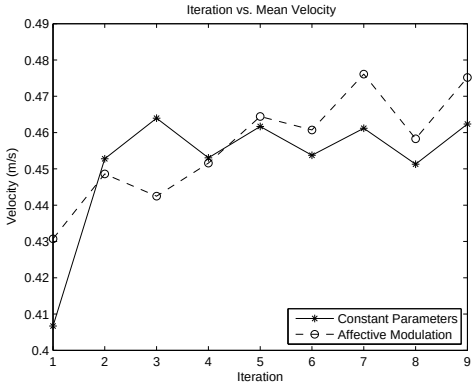

Figure 10.46: Set F – Velocity.

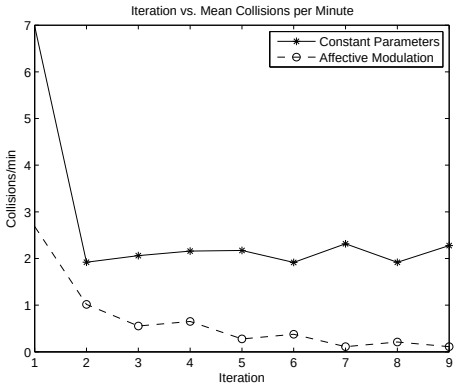

Figure 10.47: Set F – Collisions per minute.

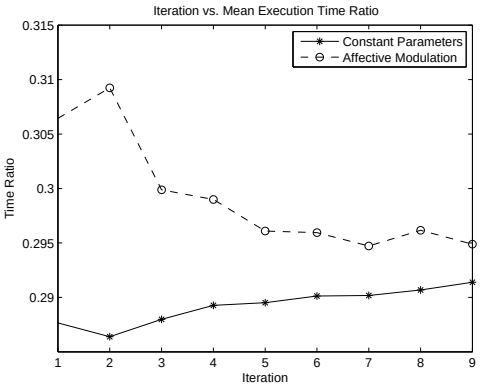

Figure 10.48: Set F – Execution time ratio.

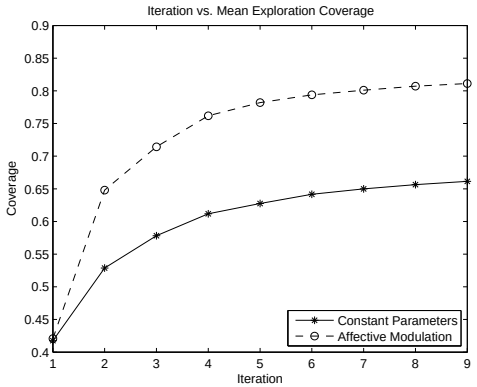

Figure 10.49: Set F – Exploration coverage.

10.2.7 Environment Set G

Environment set G (Figures 10.50 and 10.51) also includes unseen obstacles, but it is otherwise similar to set C. Overall, completion times (Figure 10.52), velocity (Figure

10.53), execution time ratios (Figure 10.55) and exploration coverage (Figure 10.56) show similar trends to those resulting from set C.

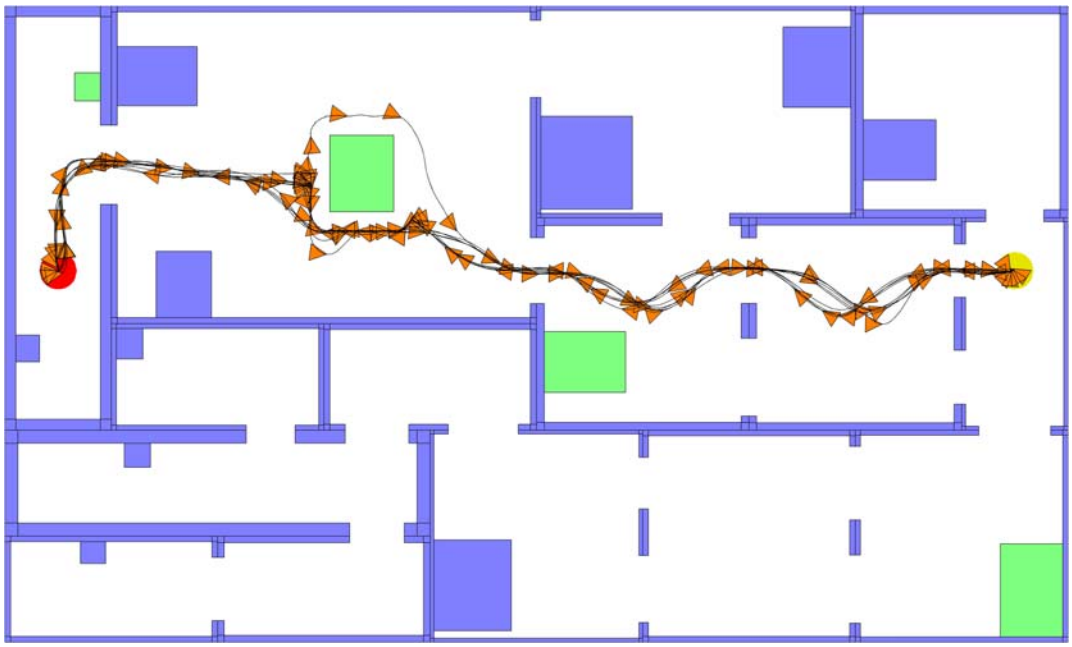

Figure 10.50: Set G – Example path resulting from constant parameters.

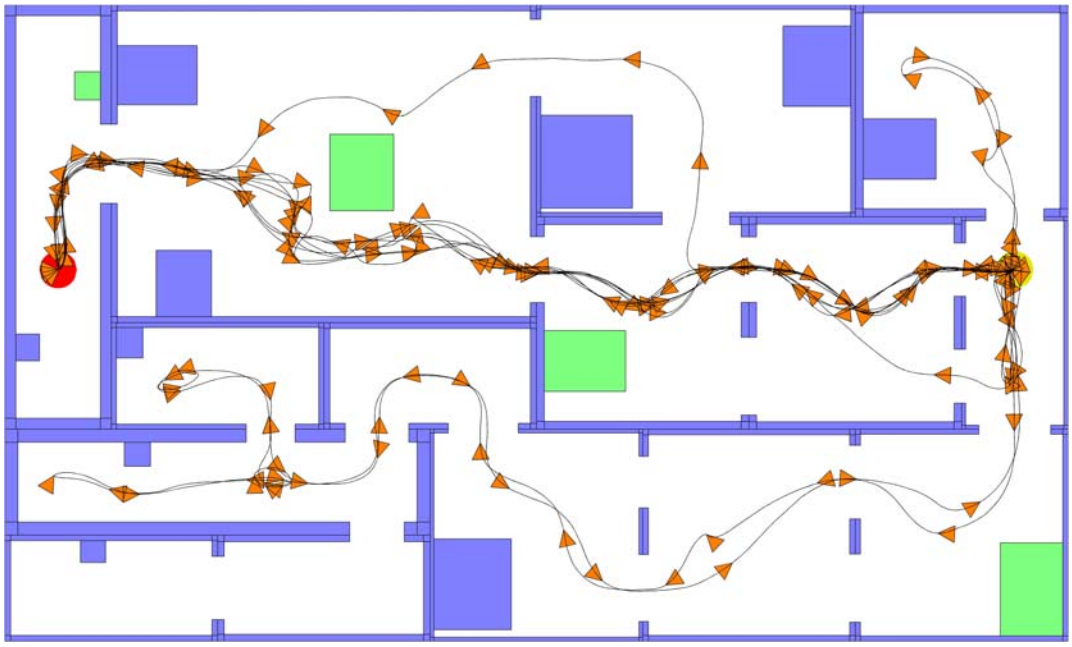

Figure 10.51: Set G – Example path resulting from affective parameter modulation.

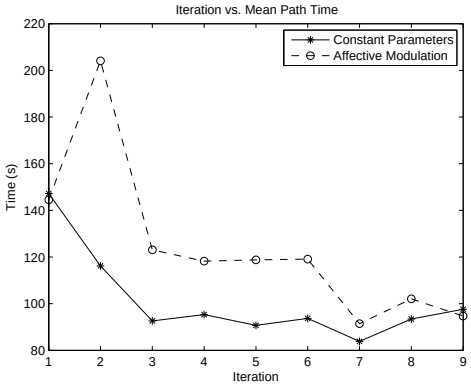

Figure 10.52: Set G – Mean completion time for 100 experimental runs per iteration.

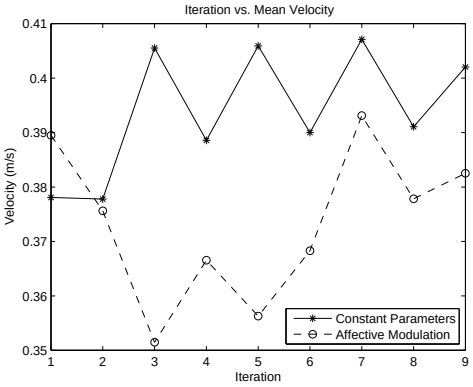

Figure 10.53: Set G – Velocity.

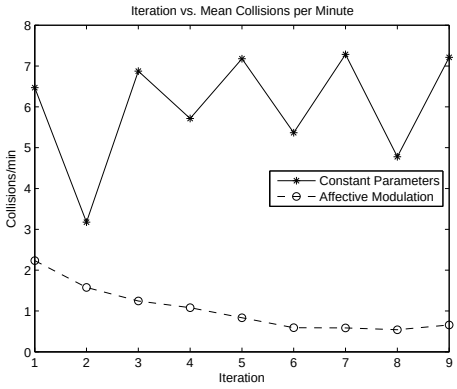

Figure 10.54: Set G – Collisions per minute.

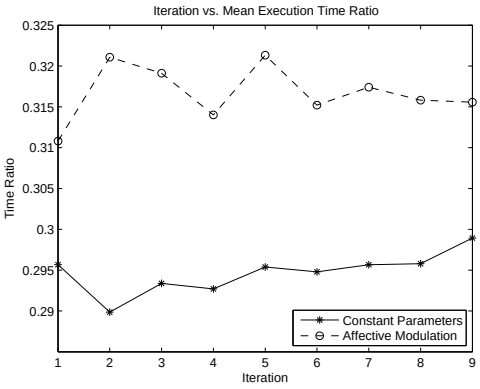

Figure 10.55: Set G – Execution time ratio.

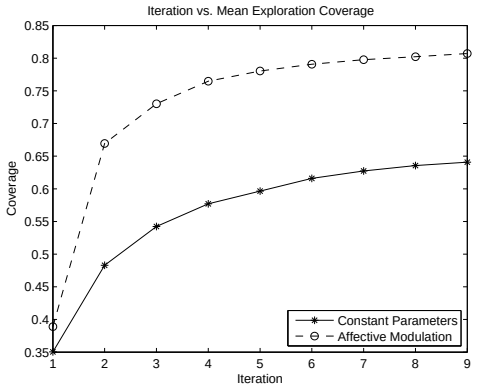

Figure 10.56: Set G – Exploration coverage.

A slightly larger gap exists between the completion times of the affective and static systems as a result of a slightly higher number of obstructed states caused by increased map-based avoidance weights. Nevertheless, they again converge towards similar levels during the final iterations. As with set F, the affect model produces a

dramatic reduction in the number of collisions (Figure 10.54), and this reduction improves consistently as the robot continues to traverse the environment.

10.2.8 Results Comparison

Results from the different environment sets are summarised in Tables 10.1 and 10.2. Performance characteristics averaged from all nine iterations are shown, as well as those from the final two iterations of each experimental run. These final iterations are relevant because they reveal the robot's capabilities after it has had sufficient opportunity to explore and adapt to the environment. Exploration coverage is a sum rather than an average of multiple iterations, so only one value is shown per environment/configuration combination.

TABLE 10.1: PERFORMANCE CHARACTERISTICS – CONSTANT PARAMETERS

Env. set	Iteration	Time (s)	Velocity (m/s)	Collisions /minute	Execution time ratio	Coverage
A	All	39	0.51	0	0.20	0.57
	Final	38	0.51	0	0.20	
B	All	68	0.46	0.0058	0.29	0.60
	Final	64	0.46	0	0.29	
C	All	105	0.41	0.079	0.30	0.67
	Final	101	0.41	0.085	0.30	
D	All	81	0.44	0.28	0.32	0.65
	Final	74	0.44	0.20	0.32	
E	All	106	0.41	0.27	0.32	0.75
	Final	94	0.41	0.29	0.32	
F	All	76	0.45	2.6	0.29	0.66
	Final	69	0.46	2.1	0.29	
G	All	101	0.39	6.0	0.29	0.64
	Final	96	0.40	6.0	0.30	

TABLE 10.2: PERFORMANCE CHARACTERISTICS – AFFECTIVE PARAMETER MODULATION

Env. set	Iteration	Time (s)	Velocity (m/s)	Collisions /minute	Execution time ratio	Coverage
A	All	33	0.59	0	0.18	0.69
	Final	32	0.62	0	0.17	
B	All	72	0.46	0	0.30	0.77
	Final	59	0.47	0	0.29	
C	All	115	0.39	0.10	0.32	0.77
	Final	100	0.39	0.14	0.33	
D	All	86	0.43	0.25	0.34	0.80
	Final	72	0.44	0.15	0.33	
E	All	108	0.40	0.27	0.34	0.82
	Final	89	0.41	0.38	0.33	
F	All	82	0.46	0.67	0.30	0.81
	Final	67	0.47	0.16	0.30	
G	All	123	0.37	1.00	0.32	0.80
	Final	98	0.38	0.60	0.32	

Overall, the benefits of affective parameter modulation outweigh its costs in five out of the seven environment sets tested (A, B, D, F and G). The two systems yield similar performance in set E, while set C favours the system with constant parameters. While different configurations of constant parameters may close the gap between the two approaches in a single environment set, the navigation requirements of different types of environments are sufficiently unique that no single set of parameters is likely to be optimal for all of them. Additional tuning may further improve the model's performance in certain situations, but the presented results give an indication of its potential as an adaptation mechanism.

10.3 Extensions

Now that the completed system has been demonstrated and tested, some extensions to this research are briefly investigated in preliminary experiments.

10.3.1 Distribution of Dataset

This thesis focuses on revealing trends in the performance data, rather than absolute quantitative values. Absolute values are largely dependent on domain-specific variables relating to the robot and its environment, whereas many trends are more general. The large number of experiments involved also makes it impractical to present a detailed statistical analysis for every result. Thus, the mean of each performance characteristic was presented rather than the full distribution or error bars. As with any summarising statistic, the mean hides some information contained within the dataset, so a more detailed analysis could be advantageous.

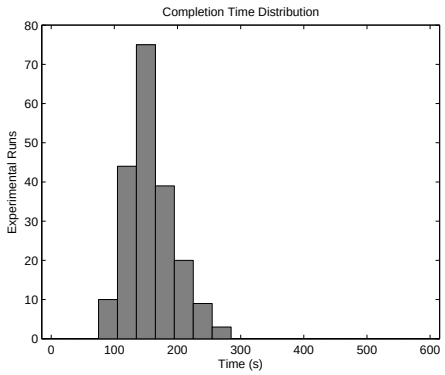

Figure 10.57: Distribution of completion times for a single environment in set C (200 experimental runs).

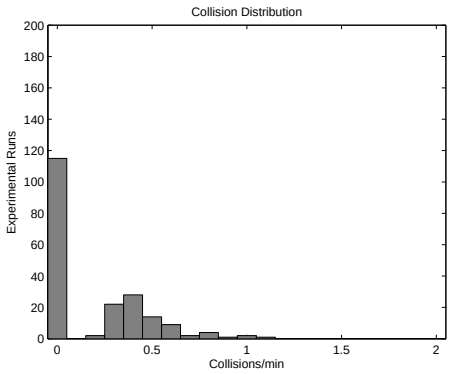

Figure 10.58: Distribution of collision rates for a single environment in set C.

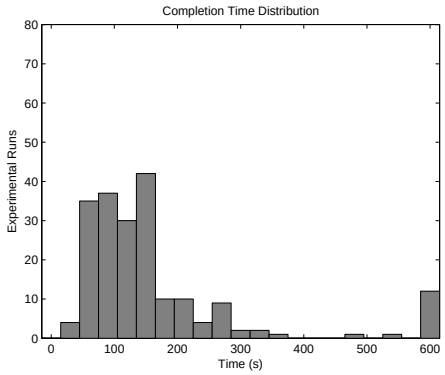

Figure 10.59: Distribution of completion times for all environments in set C.

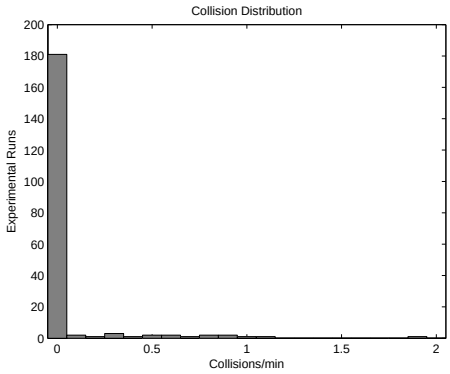

Figure 10.60: Distribution of collision rates for all environments in set C.

Figures 10.57 and 10.58 show example distributions obtained by repeating an experimental run 200 times in a single environment from set C (selected because it presents a medium-level challenge). Figures 10.59 and 10.60 show distributions

resulting from repeated experiments in the full set of 20 environments in set C (for the same total of 200 experimental runs).

A relatively narrow distribution of completion times occurs in the single environment (Figure 10.57), due to the very similar paths travelled by the robot each time (Figure 10.61). The distribution is skewed towards lower values because the robot generally makes rapid progress towards the goal, but it occasionally becomes delayed by a narrow doorway. This non-normal distribution is particularly apparent for collision rate (Figure 10.58). The likelihood of a collision occurring during a given experimental run is low, but the potential consequences of a real-world collision are severe (e.g. damage to the robot or its environment). This is one reason for choosing the mean rather than median as the primary means to summarise data (the median collision rate is zero in many experiments).

In multiple environments the results are spread over a larger range, and the distribution is again skewed towards lower values (Figures 10.59 and 10.60). Twelve experimental runs result in the robot timing out without reaching the goal. This yields a recorded completion time of 600 seconds, even though the actual completion time is effectively infinite.

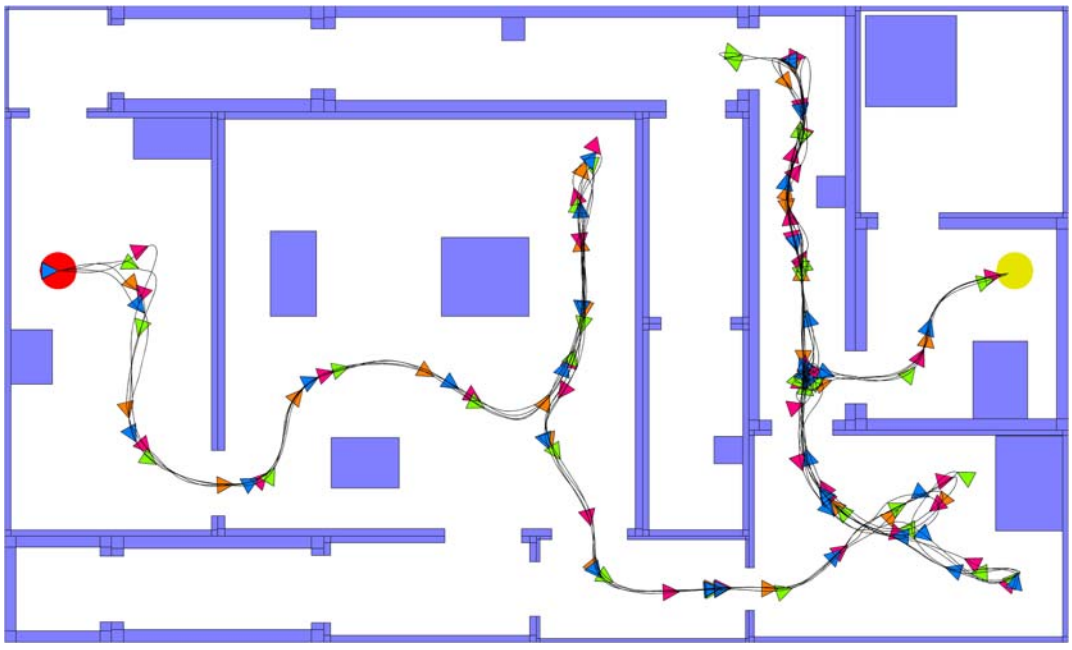

Figure 10.61: Four repeated experimental runs through an environment in set C, each represented by direction arrows of different colours.

While it is impractical to present this quantity of information for each result (since each histogram shown is reduced to single points on the graphs in this thesis), future research is likely to have a narrower focus than this thesis, and thus could benefit from a more complete representation of the dataset. Due to the non-normal distributions involved, simple standard deviations or error bars are not ideal metrics for the statistical analysis of this dataset. It may be more beneficial to present the data in its raw form (e.g. distribution histograms).

10.3.2 Human Influences

The previous navigation experiments have been conducted without considering human interactions with affective robots. The simulated humans in these experiments simply move randomly. However, real humans may be influenced by robot behaviour that they perceive as emotional. This could potentially be exploited by a robot to improve its ability to complete navigation tasks.

One example of a potentially advantageous interaction involves the emotion anger. In our implementation, the primary role of anger is to respond to perceived obstructed states by reducing safety-enhancing, but convergence-impeding behavioural tendencies. However, in environments populated by humans, anger can be assigned an additional role: communicating its difficulties to people nearby. In public venues, curious humans are likely to stand around a robot, impeding its progress. In such an event, the robot's anger will slowly build, and it could communicate its changing state through audio messages of increasing insistence (and decreasing politeness), such as:

- Low anger: "Please allow me to pass."
- Medium anger: "You're blocking the corridor."
- High anger: "Get out of my way!"

Additionally, a robot such as MARVIN can adopt a more aggressive body posture that may be more likely to compel a human to comply with its instructions.

As a preliminary test of this idea, the experiment conducted in environment set E is repeated, but the behaviour of the simulated humans is modified. If the robot's anger intensity exceeds 0.75 in this modified experiment, a simulated message is triggered, causing any humans within 5 m of the robot to attempt to move away from it. While

this response is highly simplified compared to the varied responses of real humans (who may choose to ignore an instruction to move away from the robot), it should give an indication of some of the effects of this approach on robot performance.

The resulting completion times and collision rates are compared with the previous experimental results (both with and without affective parameter modulation) in Figures 10.62 and 10.63. A tangible reduction in collisions is apparent when human responses to anger are simulated (Figure 10.62), because the robot is less likely to attempt to force its way past obstructing humans. Completion times are increased by this modification (Figure 10.63). This is likely because the presence of simulated humans actually help the robot pass through some narrow doorways, and this potential advantage has been removed.

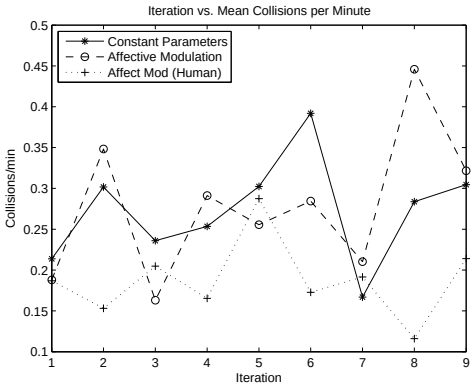

Figure 10.62: Human influences – Collisions per minute for 100 experimental runs per iteration.

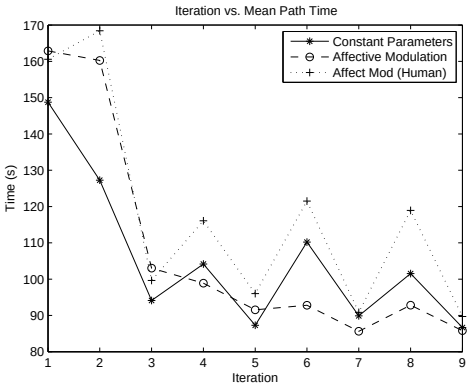

Figure 10.63: Human influences – Completion time.

Another possible application of affective influences on robot behaviour is revealed by this experiment. Instead of merely requesting that all humans move out of the way when it becomes obstructed, the robot could request a human's assistance. Most humans would intuitively understand that moving behind the robot might encourage it to pass through a doorway, while moving in front of it would block its path. Hence, this request could be as simple as: "I'm stuck. Please help." In this manner, some of the difficulties associated with navigation in populated environments could be turned into advantages.

10.3.3 Robotic Personality

Modulating a robot's underlying planning and control architecture parameters can significantly alter its behavioural tendencies. Similarly, the responses of the affect model can be altered by changing its internal parameters. Such parameters include the growth and decay rates of different affective states and the weights associated with various interconnections between them. Because they are largely time-invariant and influence a robot's 'emotional' behaviour, these parameters can be regarded as analogous to biological personality traits.

Robotic models of affect generally do not include explicitly defined personality traits, with the exception of the TAME framework (Moshkina and Arkin, 2005), which employs a taxonomy based on the Five-Factor Model of Personality (McCrae and Costa, 1992). Personality traits based on the Five-Factor Model can be related to various affective states included in our model (shown in Table 10.3).

TABLE 10.3: PERSONALITY TRAITS

Personality trait	**Human tendencies**	**Affective weights**
Openness	Intellectual curiosity, acceptance of novelty	Exploration
Agreeableness	Compassion, cooperation with others	N/A
Conscientiousness	Self-discipline, ambition, planning	Action, introspection
Extraversion	Positive emotions, to seek stimulation and social interaction	Positive mood
Neuroticism	Negative emotions	Negative mood

Two personality traits that may be useful in our system are extraversion and neuroticism. Extraversion can be represented by the weight controlling the intensity of positive moods, while neuroticism can be represented by the equivalent weight for negative moods. Two different weight combinations are tested in environment set C, resulting in the paths and mood/drive plots shown in Figures 10.64-10.67. The first configuration (Figures 10.64 and 10.66) represents high neuroticism and low extraversion, while the second (Figures 10.65 and 10.67) represents low neuroticism and high extraversion. The 'neurotic' personality quickly escapes from obstructed

states by lowering its safety drive. Conversely, the ‘extraverted’ personality is relatively ineffective in this environment, as it is unable to respond adequately to these obstructed states.

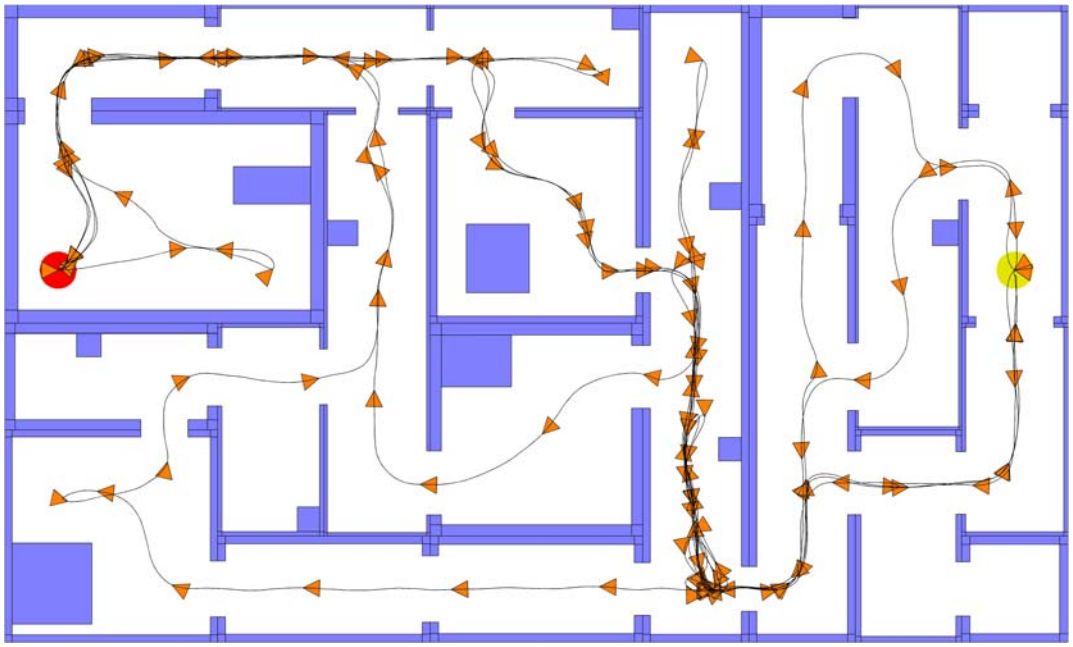

Figure 10.64: Path travelled by a ‘neurotic’ robot (negative mood weight = 1.5, positive mood weight = 0.5).

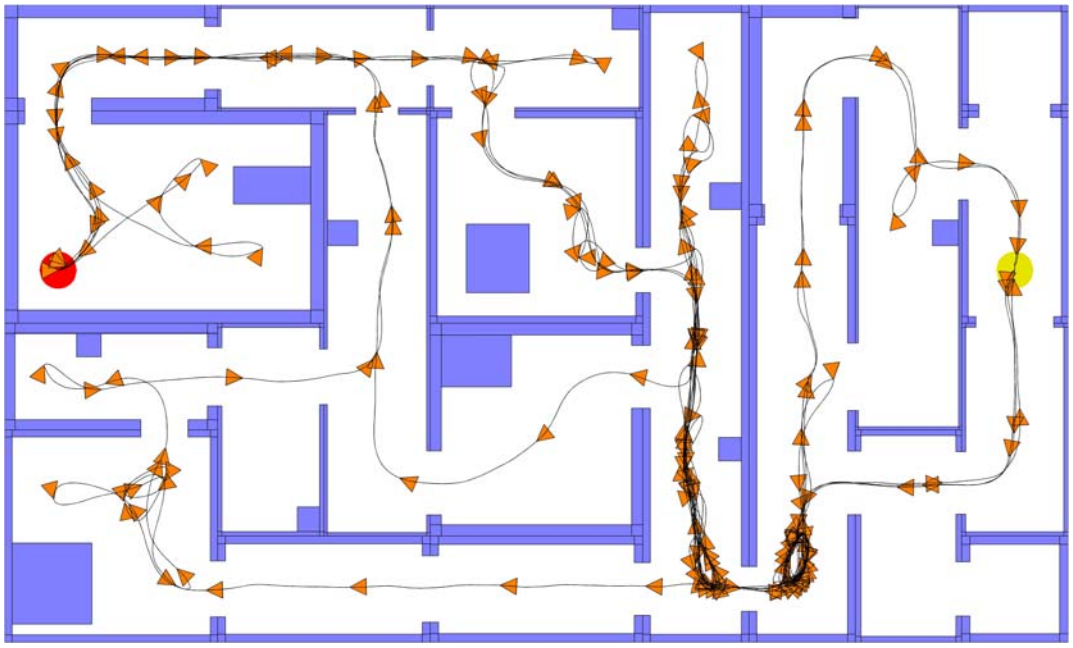

Figure 10.65: Path travelled by an ‘extraverted’ robot (negative mood weight = 0.5, positive mood weight = 1.5).

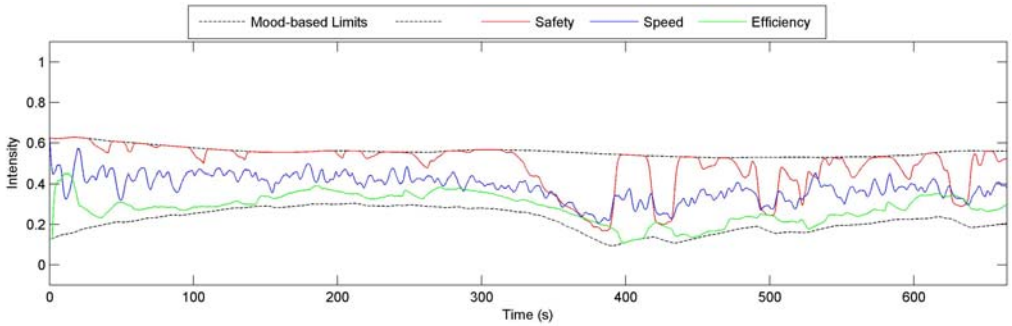

Figure 10.66: Moods and survival drives of 'neurotic' robot (negative mood weight = 1.5, positive mood weight = 0.5).

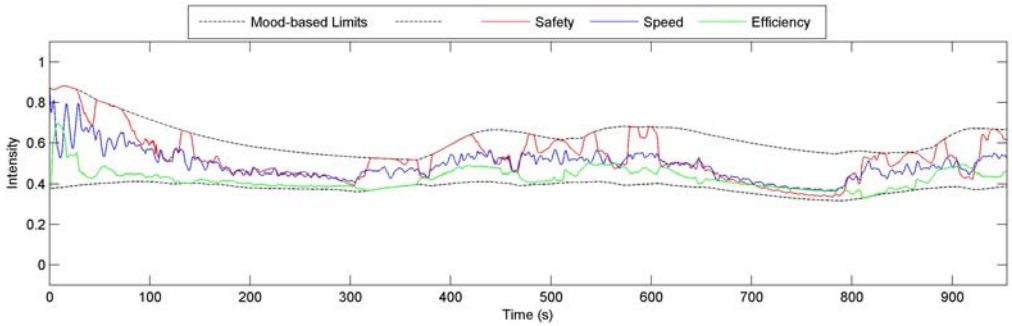

Figure 10.67: Moods and survival drives of 'extraverted' robot (negative mood weight = 0.5, positive mood weight = 1.5).

This simple experiment does not reveal much about the system that is not already known, but it could be useful to represent these weights and other parameters to personality traits, particularly if the model is extended into other domains such as human-robot interaction and multi-robot systems.

10.4 Summary

This chapter presented experimental results obtained from the fully-integrated affective planning and control architecture. Dynamically changing intensities of different affective states were demonstrated, showing their behaviour in example environments. The affective system was compared to an equivalent system operating with constant parameters. To thoroughly assess the performance contributions of the affect model in different situations, these experiments were repeated in a wide range of different environment sets. Results indicate that the affective system outperforms its static equivalent in a majority of test scenarios.

Experiments involving several extensions to this research were also conducted. First, we presented a more complete statistical distribution of results for experiments in single and multiple environments. Next, we investigated the effects of feedback between robotic emotions, human responses and robot performance. Finally, a brief investigation of possible explicit representations of robotic personality traits in our model was conducted.

11 Conclusions

The concepts of affect and emotion have traditionally been associated with sociable robotics, because many of the social aspects of emotion are clearly apparent or implicitly understood. In contrast, this thesis described the development and implementation of a mobile robot control architecture that incorporates emotion and other affective states as general adaptation mechanisms to improve its performance in a non-social context. Affect is not the dominant motivator of a robot's actions, but rather, a secondary influence that continuously modulates the decisions and actions of its cognitive systems.

11.1 Planning and Control System

Cognition is implicitly represented by a hybrid reactive/deliberative planning and control system. This controller is applied to the problem of mobile robot navigation and exploration in arbitrary flat-surfaced environments. Three distinct control layers are modelled, and each layer formulates the problem in a different conceptual space.

Deliberative mapping and path planning systems operate in a discretised Cartesian space representing the robot's global environment. A rectangular occupancy grid map is updated in real-time based on each node's proximity to sensor beams, obstacles and the robot. Path planning is accomplished by an A* algorithm (Judea, 1984) modified to incorporate continuous cost values derived from multiple fused grid maps (e.g. occupancy, exploration, danger and emotions) rather than binary occupied/free-space representations sorted by thresholds. By incorporating a number of different grid maps into its deliberative systems, the planner can take into consideration goals other than simply finding the shortest path between two points.

Next, the local environment is represented as a polar histogram inspired by the vector field histogram (VFH) obstacle avoidance method (Borenstein and Koren, 1991), and a locally optimal heading is obtained for the robot. Alternative navigation strategies emerge from different weighted combinations of objectives such as sensor-based obstacle avoidance, map-based avoidance, path following and reactive goal seeking. Unlike traditional behaviour-based approaches (e.g. Brooks, 1986), this approach allows competing objectives to simultaneously contribute to decision making, and

their relative contributions can be smoothly adapted to suit a given task and environment.

Finally, inspired by velocity space obstacle avoidance approaches such as the curvature velocity (Simmons, 1996) and dynamic window (Fox et al., 1997) methods, the local navigation problem is applied to a rectangular space containing candidate linear and angular velocities. This search space is bounded by the minimum and maximum velocities achievable given the robot's kinematic and dynamic constraints. Outputs are again selected by incorporating weighted contributions of multiple objectives.

The navigation approaches that inspire this system typically focus on computational efficiency and robustness. Thresholds and hard constraints are often employed to limit computational complexity and to prevent a robot from performing actions that are expected to result in a collision. Redundant systems may be viewed in a negative light due to their increased potential for conflicts and squandered CPU cycles. In contrast, the emphasis of this project is on flexibility and adaptation, which has yielded some innovations that may contribute to the improvement of future mobile robot navigation algorithms.

This system integrates two distinct reactive navigation styles – directional and velocity space control – in a manner that differs from previous implementations. The lane-curvature method (Ko and Simmons, 1998) and beam-curvature method (Benayas et al., 2002) are both examples that incorporate directional and velocity space elements, but they are essentially implemented as single layers. In contrast, the directional and velocity controllers of our system are modelled as distinct stages that can to some extent operate independently when required. Results indicate that combining the two approaches in this manner can be beneficial to performance, as they possess complementary strengths and weaknesses. The directional controller provides superior goal-directed performance, while the velocity controller greatly reduces the likelihood of high-speed collisions.

The original directional and velocity space approaches had little in common. Some progress has been made towards unifying them so that they can be more easily compared and combined. To that end, both layers employ weighted product objective functions of the same basic form. In principle, this objective function offers some

advantages over the weighted sum functions utilised approaches such as VFH+ (Ulrich and Borenstein, 1998), curvature velocity (Simmons, 1996) and dynamic window (Fox et al., 1997). It helps negate the possibility of the robot choosing a highly unfavourable option (for example, prioritising goal seeking over obstacle avoidance when such an action would result in a collision), but it nevertheless allows even low-priority objectives to influence the robot's behaviour.

The underlying control architecture developed for this research has subsequently been utilised in two other projects. Doctoral candidate Praneel Chand implemented an early version on multiple heterogeneous real-world and simulated mobile robots for his research on multi-robot coordination (Lee-Johnson et al., 2007). Some of these robots possess differential drive systems similar to the simulated version of MARVIN utilised in this research, but the architecture was also successfully applied to a pair of real-world tricycle mobile robots that steer by rotating their rear wheels. Masters student Thomas Roehr implemented a more recent version on a simulated mobile robot intended for research on urban search and rescue applications (Roehr, 2008). Chand utilised a MATLAB simulator based on the one outlined in this thesis and developed a real world robot controller in C++, whereas Roehr employed the commercial simulator Microsoft Robotics Studio. The implementation of this approach on multiple different hardware and software platforms is an important validation of its suitability as a general control approach for wheeled mobile robots.

11.2 Affective Navigation

Being highly flexible, this controller enables the robot to adjust its navigation strategies to suit its environment, task, momentary situation and the resources at its disposal. This flexibility arises from numerous weights and other parameters that can be adapted to modify the robot's behaviour either subtly or overtly as required. Results show that there are many situations where the robot must make tradeoffs between competing performance characteristics. No single set of parameters can yield optimal behaviour in all conceivable environments and situations. Instead, planning and control parameters are modulated by the affect model to facilitate behavioural changes that are likely to be beneficial to performance.

This model is inspired by theories proposed by authors such as Dörner and Hille (1995) that represent emotions as continuous parameter modulations rather than cognitive epiphenomena or discrete states in a state machine. Affective stimuli, emotions, moods and drives are modelled as continuous intensities that interact to produce these parameter modulations. Stimuli are derived from various sensor and representation data. Each stimulus represents an internal or external event such as an obstructed state or collision for which an affective response is deemed advantageous by the system designer. The stimuli are combined by a dynamic weighted sum function to form a small set of basic emotions, based on theories that some biological emotions are distinct and hardwired, rather than cultural artefacts (e.g. Ekman, 1992). Activated emotions are functions of both current stimuli and previous emotions elicited near the robot's current location. These previous emotions are stored in grid maps, which are also passed to the deliberative controller as location-specific biases to path-planning. Current emotion intensities control a set of drives, each of which is responsible for modulating a subset of related control parameters. Drive intensities are constrained by moods, changing the degree of influence that some affective processes exert over the robot's behaviour.

The development and implementation of this model has yielded several contributions to the emerging field of emotion-based robotic control. Although some existing models share certain similarities to ours, none resemble it in its entirety. The domain to which the model is applied (mobile robot navigation) is very different from that of the majority of robotic implementations, which tend to focus on social aspects of emotion and affect. The few implementations that apply affect to mobile robot controllers in a largely non-social context (e.g. Tingley and Browne, 2006; Neal and Timmis, 2003; Gadanho and Hallam, 2001) typically employ simple (reactive) behaviour-based architectures and/or simple affective interactions (e.g. emotion-driven action selection). In contrast, the architecture to which this model is applied is a robust multilayered system capable of successful navigation and exploration in complex environments even without the aid of emotional influences. The affect model is assigned the ambitious task of improving the performance of this already well-optimised system.

An important focus of this research involved conducting experiments in a range of procedurally-generated environments to assess the quantitative performance

contributions of affective parameter modulation. Results of these experiments indicate several situations in which this approach can improve performance. The affective robot can adjust the amount of computational effort applied to a navigation task depending on its perceived difficulty. It can vary its speed and/or safety parameters in response to the navigational challenges it encounters. In environments where its map is inaccurate or its sensors cannot always be relied upon, the robot can adjust the bias between reactive and deliberative control approaches and/or environment representations. Affect also adapts the robot's incentive to explore its environment while it navigates from point to point.

While no claims are made that the model outperforms other adaptive control approaches in these individual areas, this thesis shows that it can be successfully applied to a broad range of control problems. Other adaptive mobile robot controllers (the majority of which are not inspired by emotions or affect) typically focus on a narrower set of problems and are applied to much simpler underlying controllers (e.g. purely reactive schema-based methods).

From this research, one can speculate on some of the general functional roles of emotion and affect in intelligent systems. One of these roles is behaviour prioritisation, or assigning higher priority to behaviours that are likely to improve performance or survival. While some computational architectures employ a simple switch that is triggered when a parameter crosses some threshold, it can be beneficial to model smoother transitions such as the shifting weights utilised in our architecture. Another related function is resource management, or limiting the cognitive resources applied to a problem so that they do not exceed those required for the task at hand. In systems where behaviours are represented as binary influences (activated/deactivated), this may simply involve disabling unused behaviours. However, in architectures such as ours where competing behaviours are simultaneously activated, such an approach is less feasible. Instead, the computational complexity of lower priority subsystems should be reduced in a continuous manner, e.g. by reducing the resolutions of their search spaces.

Modulatory affective influences can be regarded as a form of 'intelligent noise', or as a compromise between deterministic and stochastic behaviour. Like random noise, affective mechanisms can help an intelligent system escape from some repetitive cycles. Increased response variation may yield increased flexibility, allowing an

affective system to remain functional when presented with unexpected circumstances. Unlike purely random noise, affective mechanisms can be selectively biased towards responses that in the long term are beneficial to survival and goal completion. They are not always beneficial, but through careful implementation their potential negative effects can be reduced. This tilts the robot's behaviour back towards determinism – an emotional response can be more easily controlled than a purely random response. Nevertheless, affective responses are sometimes difficult to predict and control. Intelligent system designers must inevitably make trade-offs between flexibility and stability, and affect may improve the former at a cost to the latter.

11.3 Future Work

During the navigation experiments described in this thesis, the robot was assumed to know its location at all times. If the system were demonstrated in long-term experiments on a real-world mobile robot without the aid of external systems such as GPS or active landmarks, odometry errors would accumulate to the point where it fails to converge on the goal point. Figure 11.1 shows a distorted grid map generated during a single traversal of a simulated environment with systematic odometry errors enabled. Subsequent traversals soon render the controller non-functional. Hence, it could benefit from the implementation of a simultaneous localisation and mapping (SLAM) algorithm (for a review of various successful approaches, see Durrant-Whyte and Bailey, 2006). Furthermore, emotion-influenced localisation may be an interesting and fruitful avenue for future research.

One advantage of the planning and control approach employed in this research is that each layer is highly extensible. The deliberative layer can be extended by incorporating additional maps into the path planning process. Task-specific areas of strategic importance can be positively or negatively biased to encourage or dissuade the robot from planning paths through them. For a security robot such as MARVIN, biased regions could represent efficient patrol routes or known hotspots for unauthorised activities. The reactive layers can each be extended by adding new objectives to their objective functions. This is particularly useful to the directional controller, which can simultaneously pursue multiple paths and/or goal points, assigning different weights to each.

The map-based avoidance functions implemented in both the directional and velocity control layers presently utilise thresholds to mark map nodes as either occupied or unoccupied. They could be improved if this binary representation were replaced with a more continuous approach that takes into consideration subtler differences between node costs. This may help prevent certain obstructions that currently occur, and it could also allow the reactive layers to respond to location-specific emotional influences (e.g. avoiding regions associated with sadness).

Figure 11.1: Example occupancy grid map generated when a systematic error of 0.5% is applied to the robot's right odometer.

Connections between the various affective components of our architecture are hardwired. A simple form of learning is implemented in the form of emotion maps that are updated over time, but the robot cannot adapt its responses to a given emotion or mood. The input-output mappings produced by the dynamic weighted sum function can be represented by equivalent Sugeno fuzzy inference systems (Sugeno, 1985). Although less computationally efficient than the current implementation, they can be trained by neuro-adaptive learning techniques. Future investigations of these and other computational mechanisms to adaptively connect and combine affective components may yield further insights into the interactions between affect and learning.

Preliminary experiments involving two other possible extensions to this research were described in Chapter 10. An experiment was conducted investigating the navigation performance effects of simulated human responses to the portrayal of robotic anger.

Further experiments could be performed to examine the influences of other emotions and human reactions. Given the unpredictable nature of real human responses, simulations are no substitute for real-world experiments in populated environments, but this was not feasible due to time and resource constraints.

Another experiment briefly illustrated the behaviour of different 'robotic personalities', represented by distinct affective parameter configurations. Explicitly defined personality traits are of limited value to an individual robot in a non-social context, but they may become useful if this research is extended to other domains. In humans, different personality types are generally suited to different roles, so the inclusion of a diverse range of personalities may improve the long-term survival prospects of a community. Hence, explicit representations of personality traits could be well-suited to the domain of multi-robot systems, where distinct personalities could be tailored or evolved to suit each robot's role within the group. Similarly, social robots could be assigned personality traits that suit individual humans with whom they interact.

11.4 International Publications

A paper describing an earlier version of the underlying navigation system implemented in this project (and in the multi-robot coordination project undertaken by Praneel Chand) was presented in:

- C.P. Lee-Johnson, P. Chand, and D.A. Carnegie, "Applications of an adaptive hierarchical mobile robot navigation system," *Proc. Australasian Conference on Robotics and Automation*, Brisbane, Australia, 2007. To be reprinted in *Amazing Strides in Robotics*, ICFAI University Press.

Several international conference/journal publications have emerged from the research on affective parameter modulation of mobile robot controllers:

- C.P. Lee-Johnson and D.A. Carnegie, "Towards a computational model of affect for the modulation of mobile robot control parameters," *Proc. Third International Conference on Autonomous Robots and Agents*, pp 367-372, Palmerston North, New Zealand, December 2006.

- C.P. Lee-Johnson and D.A. Carnegie, “Emotion-based parameter modulation for a hierarchical mobile robot planning and control architecture,” *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2839-2844, San Diego, CA, 2007.
- C.P. Lee-Johnson and D.A. Carnegie, “Mobile robot navigation modulated by artificial emotions,” *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics* (in press).
- C.P. Lee-Johnson and D.A. Carnegie, “Robotic emotions: navigation with feeling,” *Handbook of Research on Synthetic Emotions and Sociable Robotics: New Applications in Affective Computing and Artificial Intelligence*, J. Vallverdú and D. Casacuberta, Eds. Information Science Reference (in press).

11.5 Summary

Original contributions made by this thesis include:

- Directional and velocity space control approaches were employed as distinct layers that each contribute to reactive control, unlike existing methods which typically employ a single reactive control layer.
- The weighted product objective function employed by the reactive layers is a potential improvement over existing weighted sum objective functions.
- Some thresholds employed by existing controllers such as VFH and curvature velocity are replaced by weights and continuous parameters, facilitating adaptation at a cost to computational efficiency.
- The application of artificial emotions and affect to mobile robot navigation differs from most models, which are typically applied in the social domain.
- Rather than modelling emotions as discrete states within a state machine, as do many existing models, they are secondary behavioural influences represented in our model as parameter modulations.
- Even disregarding the affective labels employed by our model, parameter modulation of the multilayered controller outlined in this thesis is an original

contribution. The underlying controller is significantly more complex than schema-based/potential field methods traditionally employed by adaptive controllers, but it retains the adaptive potential of those methods.

The research undertaken for this thesis has contributed to a growing body of evidence in support of the broader argument that affective computing is not merely beneficial to human-machine interaction, but also to the general problem of synthesizing adaptive behaviour. By modelling affective processes as parameter modulations, system designers can merge the robustness of traditional 'cognitive' systems with the adaptive properties attributed to biological emotions. The effectiveness of this approach as a mobile robot control mechanism also supports the work of authors such as Fellous (1999) who have advocated that human emotions may be represented as continuous patterns of neuromodulations.

Furthermore, this thesis has demonstrated some practical innovations that may be utilised to improve mobile robot planning and control systems. The focus of this research was on designing emotion-like mechanisms that are beneficial to a mobile robot controller, so extensive quantitative results have been gathered to verify the approach. In many experimental scenarios, a controller modulated by affect outperformed one operating without affective influences. These results show that emotion-like mechanisms can be successfully utilised to adapt a mobile robot controller to different environments and situations. Affective systems such as the ones demonstrated in this thesis may be crucial components of future general-purpose robotic systems, enabling them to operate safely and efficiently in highly complex, partially-observable real-world environments.

References

(Arkin, 1989) R.C. Arkin, "Towards the unification of navigational planning and reactive control," *Proc. AAAI Spring Symposium on Robot Navigation*, pp. 1-5, Menlo Park, CA, 1989.

(Arkin, 2005) R.C. Arkin, "Moving up the food chain: motivation and emotion in behavior-based robots," in *Who Needs Emotions: The Brain Meets the Robot*, J. Fellous and M. Arbib, Eds. New York, NY: Oxford University Press, 2005, pp. 425-270.

(Aurenhammer, 1991) F. Aurenhammer, "Voronoi diagrams: A survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345-405, 1991.

(Avila-García and Cañamero, 2005) O. Avila-García and L. Cañamero, "Hormonal modulation of perception in motivation-based action selection architectures," *Proc. AISB'05 Convention*, vol. 2, pp. 9-16, Hertfordshire, UK, 2005.

(Aylett, 2006) R.S. Aylett, "Emotion as an integrative process between non-symbolic and symbolic systems in intelligent agents," *Proc. AISB'06 Symposium of Brain and Mind*, pp. 43-47, Bristol, England, 2006.

(Bach, 2003) J. Bach, "The MicroPsi agent architecture," *Proc. 5th International Conference on Cognitive Modeling*, pp. 15-20, Bamberg, Germany, 2003.

(Bard, 1928) A.P. Bard, "A diencephalic mechanism for the expression of rage with special reference to the sympathetic nervous system," *American Journal of Physiology*, vol. 84, no. 1, pp. 490-515, 1928.

(Bates, 1994) J. Bates, "The role of emotion in believable agents," *Communications of the ACM*, vol. 37, no. 1, pp. 122-125, 1994.

(Baxter and Murray, 2002) M. Baxter and E. Murray, "The amygdala and reward," *Nature Reviews Neuroscience*, vol. 3, no. 7, pp. 563-573, 2002.

(Beetz and Belker, 2000) M. Beetz and T. Belker, "Autonomous Environment and Task Adaptation for Robotic Agents," *Proc. 14th European Conference on Artificial Intelligence*, pp. 648-652, Berlin, Germany, 2000.

(Benayas et al., 2002) J.A. Benayas, J.L. Fernández, R. Sanz, and A.R. Diéguez, "The beam-curvature method: A new approach for improving local real-time obstacle avoidance," *Proc. 15th IFAC World Congress*, Barcelona, Spain, 2002.

(Borenstein and Koren, 1991) J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 1, pp. 278-288, 1991.

(Breazeal, 2004) C. Breazeal, "Function meets style: insights from emotion theory applied to HRI," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 34, no. 2, pp. 187-194, 2004.

(Brock and Khatib, 1999) O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," *Proc. IEEE International Conference on Robotics and Automation*, pp. 341-346, Detroit, MI, 1999.

(Broekens, 2007) J. Broekens, "Emotion and reinforcement: Affective facial expressions facilitate robot learning," *LNAI Special Volume on AI for Human Computing*, pp. 113-132, Springer, 2007.

(Brooks, 1986) R.A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14-23, 1986.

(Buchanan and Adolphs, 2002) T.W. Buchanan, and R. Adolphs, "The role of the human amygdala in emotional modulation of long-term declarative memory," in *Emotional Cognition*, S.C. Moore and M. Oaksford, Eds. Amsterdam: John Benjamins Publishing, 2002, pp. 9-34.

(Canny, 1988) J.F. Canny, *The Complexity of Robot Motion Planning*, Cambridge, MA: MIT Press, 1988.

(Carnegie et al., 2004) D.A. Carnegie, A. Prakash, C. Chitty, and B. Guy, "A human-like semi autonomous mobile security robot," *Proc. 2nd International Conference on Autonomous Robots and Agents*, pp. 64-69, Palmerston North, NZ, 2004.

(Carroll and Russell, 1996) J.M. Carroll and J.A. Russell, "Do facial expressions signal specific emotions? Judging emotion from the face in context," *Journal of Personality and Social Psychology*, vol. 70, no. 1, pp. 205-18, 1996.

(Cawley and Carnegie, 2005) C.L. Cawley, and D.A. Carnegie, "The modification of a multi-terrain robotic vehicle for autonomous outdoor applications," *Proc. 12th New Zealand Electronics Conference*, pp. 123-128, Manukau City, NZ, 2005.

(Connell, 1992) J.H. Connell, "SSS: A hybrid architecture applied to robot navigation," *Proc. IEEE International Conference on Robotics and Automation*, pp. 2719-2724, Los Alamitos, CA, 1992.

(Damasio, 1994) A. Damasio, *Descartes' Error: Emotion, Reason, and the Human Brain*. New York, NY: G.P. Putnam, 1994.

(Damasio, 1999) A. Damasio, *The Feeling of What Happens.* London: Random House, 1999.

(Davidson, 2001) R.J. Davidson, "The neural circuitry of emotion and affective style: Prefrontal cortex and amygdala contributions," *Social Science Information*, vol. 40, no. 1, pp. 11-37, 2001.

(Dawkins, 2000) M.S. Dawkins, "Animal minds and animal emotions," *American Zoologist*, vol. 40, no. 6, pp. 883-888, 2000.

(Dörner and Hille, 1995) D. Dörner and K. Hille, "Artificial souls: Motivated emotional robots," *Proc. IEEE International Conference on Systems, Man and Cybernetics*, pp. 3828-3832, Vancouver, BC, Canada, 1995.

(Dörner, 2003) D. Dörner, "The mathematics of emotions," *Proc. 5th International Conference on Cognitive Modeling*, pp. 75-79, Bamberg, Germany, 2003.

(Durrant-Whyte and Bailey, 2006) H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *Robotics and Automation Magazine*, vol. 13, no. 1, pp. 99-110, 2006.

(Ekman et al., 1982) P. Ekman, W.V. Friesen, and P.C. Ellsworth, "What emotion categories or dimensions can observers judge from facial behavior?" in *Emotion in the Human Face* (2nd Ed.), P. Ekman, Ed. Cambridge: Cambridge University Press, 1982, pp. 39-55.

(Ekman, 1992) P. Ekman, "An argument for basic emotions," *Cognition and Emotion*, vol. 6, no. 1, pp. 169-200, 1992.

(Ekman, 1993) P. Ekman, "Facial expression and emotion," *American Psychologist*, vol. 48, no. 1, pp. 384-392, 1993.

(Elliott, 1992) C. Elliott. *The Affective Reasoner: A Process Model of Emotions in a Multi-agent System*. PhD dissertation (Technical Report 32), Northwestern University, The Institute for the Learning Sciences, 1992.

(Esau et al., 2007) N. Esau, L. Kleinjohann, and B. Kleinjohann, "Integration of emotional reactions on human facial expressions into the robot head MEXI," *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 534-541, San Diego, USA, 2007.

(Fellous, 1999) J.M. Fellous, "The neuromodulatory basis of emotion," *Neuroscientist*, vol. 5, no. 5, pp. 283-294, 1999.

(Fellous, 2004) J.M. Fellous, "From human emotions to robot emotions," *Proc. 2004 AAAI Spring Symposium: Architectures for Modeling Emotions: Cross-Disciplinary Foundations*, pp. 37-47, Palo Alto, CA, 2004.

(Fox et al., 1997) D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23-33, 1997.

(Gadanho and Hallam, 2001) S.C. Gadanho and J. Hallam, "Robot learning driven by emotions," *Adaptive Behavior*, vol. 9, no. 1, pp. 42-64, 2001.

(Gadanho, 2003) S.C. Gadanho, "Learning behavior-selection by emotions and cognition in a multi-goal robot task," *Journal of Machine Learning Research*, vol. 4, no. 1, pp. 385-412, 2003.

(Gat, 1997) E. Gat, "On three-layer architectures," in *Artificial Intelligence and Mobile Robots*, D. Kortenkamp, R.P. Bannasso, and R. Murphy, Eds. MIT/AAAI Press, 1997, pp. 195-210.

(Gelati et al., 1997) D. Galati, R. Miceli and B. Sini, "Voluntary facial expression of emotion: comparing congenitally blind with normal sighted encoders," *Journal of Personality and Social Psychology*, vol. 73, no. 1, pp. 1363-1379, 1997.

(Goel et al., 1997) A. Goel, E. Stroulia, Z. Chen, and P. Rowland, "Model-based reconfiguration of schema-based reactive control architectures," *Proc. AAAI Fall Symposium on Model-Based Autonomy*, pp. 23-39, 1997.

(Gratch and Marsella, 2004) J. Gratch and S. Marsella, "A domain-independent framework for modeling emotion," *Journal of Cognitive Systems Research*, vol. 5, no. 4, pp. 269-306, 2004.

(Gray, 1982) J.A. Gray, *The Neuropsychology of Anxiety*. Oxford: Oxford University Press, 1982.

(Gunther, 2004) Y. Gunther, "The phenomenology and intentionality of emotion," *Philosophical Studies*, vol. 117, no. 1-2, pp. 43-55, 2004.

(Haikonen, 2003) P. Haikonen, *The Cognitive Approach to Conscious Machines*, UK: Imprint Academic, 2003.

(Hess, 1957) W.R. Hess, *The Functional Organization of the Diencephalon*, London, New York: Gune & Stratton, 1957.

(Hollinger et al., 2006) G.A. Hollinger, Y. Georgiev, A. Manfredi, B.A. Maxwell, Z.A. Pezzementi, and B. Mitchell, "Design of a social mobile robot using emotion-based decision mechanisms," *Proc. IEEE International Conference on Intelligent Robots and Systems*, pp. 3093-3098, Beijing, China, 2006.

(Huq et al., 2008) R. Huq, G.K.I. Mann and R.G. Gosine, "Mobile robot navigation using motor schema and fuzzy context dependent behavior modulation," *Applied Soft Computing*, vol. 8, no. 1, pp. 422-436, 2008.

(Hwang and Ahuja, 1992) Y. Hwang, N. Ahuja, "Gross motion planning: A survey," *ACM Computing Surveys*, vol. 24, no. 3, pp. 219-291, 1992.

(Isbell, 2004) L.M. Isbell, "Not all happy people are lazy or stupid: Evidence of systematic processing in happy moods," *Journal of Experimental Psychology*, vol. 40, no. 3, pp. 341-349, 2004.

(Izard, 1971) C.E. Izard, *The Face of Emotion.* NY: Appleton-Century-Crofts, 1971.

(Izard, 1993) C.E. Izard, "Four systems for emotion activation: Cognitive and noncognitive processes," *Psychological Review*, vol. 100, no. 1, pp. 68-90, 1993.

(Judea, 1984) P. Judea, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Reading, MA: Addison-Wesley, 1984.

(Keltner and Haidt, 1999) D. Keltner and J. Haidt, "Social functions of emotions at multiple levels of analysis," *Cognition and Emotion*, vol. 13, no. 5, pp. 505-522, 1999.

(Kohler et al., 2000) C.G. Kohler, W. Bilker, M. Hagendoorn, R.E. Gur, and R.C. Gur, "Emotion recognition deficit in schizophrenia: association with symptomatology and cognition," *Biological Psychiatry*, vol. 48, no. 1, pp. 127-136, 2000.

(Ko and Simmons, 1998) N.Y. Ko, R.G. Simmons, "The lane-curvature method for local obstacle avoidance," *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1615-1621, Victoria, BC, Canada, 1998.

(Latombe, 1991) J.C. Latombe, *Robot Motion Planning*, Boston, MA: Kluwer Academic Publishers, 1991.

(Lazarus, 1991) R.S. Lazarus, *Emotion and adaptation*, NY: Oxford University Press, 1991.

(Lazarus and Folkman, 1984) R.S. Lazarus and S. Folkman, *Stress, Appraisal and Coping*. NY: Springer, 1984.

(Lee-Johnson and Carnegie, 2003) C.P. Lee-Johnson, and D.A. Carnegie, "The development of a control system for an autonomous mobile robot," *Proc. 10th New Zealand Electronics Conference*, pp. 77-82, Hamilton, NZ, 2003.

(Lee-Johnson et al., 2007) C.P. Lee-Johnson, P. Chand, and D.A. Carnegie, "Applications of an adaptive hierarchical mobile robot navigation system," *Proc. Australasian Conference on Robotics and Automation*, Brisbane, Australia, 2007.

(LeDoux, 1996) J.E. LeDoux, *The Emotional Brain*, New York, NY: Simon & Schuster, 1996.

(LeDoux, 2000) J.E. LeDoux, "Emotion circuits in the brain," *Annual Review of Neuroscience*, vol. 23, no. 1, pp. 155-184, 2000.

(McCrae and Costa, 1996) R.R. McCrae and P.T. Costa, "Toward a new generation of personality theories: Theoretical contexts for the five-factor model," *Five-Factor Model of Personality: Theoretical Perspectives*, J.S. Wiggins, Ed. New York, NY: Guilford, 1996, pp. 51-87.

(Michaud, 2001) F. Michaud, J. Audet, D. Létourneau, L. Lussier, C. Théberge-Turmel, and S. Caron, "Experiences with an autonomous robot attending AAAI," *IEEE Intelligent Systems*, vol. 16, no. 5, pp. 23-29, 2001.

(Minsky, 1986) M. Minsky, *The Society of Mind*. New York, NY: Simon and Schuster, 1986.

(Minsky, 2006) M. Minsky, *The Emotion Machine*. New York, NY: Simon & Schuster, 2006.

(Moshkina and Arkin, 2005) L. Moshkina and R.C. Arkin, "Human perspective on affective robotic behavior: A longitudinal study," *Proc. IEEE International Conference on Intelligent Robots and Systems*, pp. 1444-1451, Edmonton, Canada, 2005.

(Murphy et al., 2002) R.R. Murphy, C.L. Lisetti, R. Tardif, L. Irish, and A. Gage, "Emotion-based control of cooperating heterogeneous mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 744-757, 2002.

(Neal and Timmis, 2003) M.J. Neal and J. Timmis, "Timidity: A useful mechanism for robot control?" *Informatica*, vol. 4, no. 27, pp. 197-204, 2003.

(Nilsson, 1969) N. Nilsson, "A mobile automaton: An application of artificial intelligence techniques," *Proc. 1st International Joint Conference on Artificial Intelligence*, pp. 509-520, Washington, DC, 1969.

(Oatley and Johnson-Laird, 1987) K. Oatley and P.N. Johnson-Laird, "Towards a cognitive theory of emotions," *Cognition and Emotion*, vol. 1, no. 1, pp. 29-50, 1987.

(Ögren and Leonard, 2005) P. Ögren and N. Leonard, "A convergent dynamic window approach to obstacle avoidance," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 188-195, 2005.

(Ohman and Mineka, 2003) A. Ohman and S. Mineka, “The malicious serpent: Snakes as a prototypical stimulus for an evolved module of fear,” *Current Directions in Psychological Science*, vol. 12, no. 1, pp. 5-9, 2003.

(Ortony et al., 1988) A. Ortony, G.L. Clore, and A. Collins, *The Cognitive Structure of Emotions*. Cambridge, UK: Cambridge University Press, 1988.

(Ortony and Turner, 1990) A. Ortony and T.J. Turner, “What's basic about basic emotions?” *Psychological Review*, vol. 97, no. 1, pp. 315-331, 1990.

(Panksepp, 1982) J. Panksepp, “Toward a general psychobiological theory of emotions,” *The Behavioral and Brain Sciences*, vol. 5, no. 1, no. 1, pp. 407-467, 1982.

(Parrott and Spackman, 2000) W.G. Parrott and M. Spackman, “Emotion and memory,” in *Handbook of Emotions* (2nd Ed.), M. Lewis and J. Haviland-Jones, Eds. New York, NY: Guilford, 2000, pp. 476-490.

(Parussel, 2006) K.M. Parussel, *A Bottom-up Approach to Emulating Emotions Using Neuromodulation in Agents*. PhD thesis, University of Stirling, 2006.

(Peters et al., 2001) R.A. Peters II, K. Kawamura, D.M. Wilkes, K.A. Hambuchen, T.E. Rogers, and A. Alford, “ISAC humanoid: An architecture for learning and emotion,” *Proc. IEEE-RAS International Conference on Humanoid Robots*, pp. 459-459, Tokyo, Japan, 2001.

(Picard, 1995) R. Picard, *Affective computing*, Technical Report 321, MIT Media Laboratory, 1995.

(Picard, 2003) R.W. Picard, “What does it mean for a computer to ‘have’ emotions?” in *Emotions in Humans and Artifacts*, R. Trappl, P. Petta and S. Payr, Eds. MIT Press, 2003.

(Pickel et al., 2003) K.L. Pickel, T.A. French and J.M. Betts, “A cross-modal weapon focus effect: The influence of a weapon’s presence on memory for auditory information,” *Memory*, vol. 11, no. 1, pp. 277-292, 2003.

(Plutchik, 1980) R. Plutchik, “A general psychoevolutionary theory of emotion,” in *Emotion: Theory, Research, and Experience. Vol. 1. Theories of Emotion*, R. Plutchik and H. Kellerman, Eds. New York, NY: Academic Press, 1980, pp. 3-31.

(Plutchik, 2001) R. Plutchik, "The nature of emotions," *American Scientist*, vol. 89, no. 1, pp. 344-344, 2001.

(Ram et al., 1992) A. Ram, R.C. Arkin, K. Moorman, R.J. Clark, "Case-based reactive navigation: A case-based method for on-line selection and adaptation of reactive control parameters in autonomous robotic systems," *Technical Report GIT-CC-92/57*. Atlanta, GA: College of Computing, Georgia Institute of Technology, 1992.

(Ratner, 1989) C. Ratner, "A social constructionist critique of naturalistic theories of emotion," *Journal of Mind and Behavior*, vol. 10, no. 1, pp. 211-230, 1989.

(Reisberg and Heuer, 2004) D. Reisberg and F. Heuer, "Memory for emotional events," in *Memory and Emotion*, D. Reisberg and P. Hertel, Eds. NY: Oxford University Press, 2004, pp. 3-41.

(Roehr, 2008) T.M. Roehr, *Control of a Hierarchical Ream of Robots for Urban Search and Rescue*, MSc thesis, Victoria University of Wellington, 2008.

(Rolls, 1999) E.T. Rolls, *The Brain and Emotion*, Oxford University Press.

(Russell, 1980) J.A. Russell, "The circumplex model of affect," *Journal of Personality and Social Psychology*, vol. 39, no. 1, pp. 1161-1178, 1980.

(Santamaria and Ram, 1997) J.C. Santamaria and A. Ram, "Learning of parameter-adaptive reactive controllers for robotic navigation," *Proc. World Multiconference on Systemics, Cybernetics, and Informatics*, Caracas, Venzuela, 1997.

(Schachter and Singer, 1962) S. Schachter and J.E. Singer, "Cognitive, social and physiological determinants of emotional state," *Psychological Review*, vol. 69, no. 1, pp. 379-399, 1962.

(Simmons, 1996) R. Simmons, "The curvature-velocity method for local obstacle avoidance," *Proc. IEEE International Conference on Robotics and Automation*, pp. 3375–3382, Minneapolis, MN, 1996.

(Simpson et al., 2001) J.R. Simpson, Jr., W.C. Drevets, A.Z. Snyder, D.A. Gusnard, and M.E. Raichle, "Emotion-induced changes in human medial prefrontal cortex: II. During anticipatory anxiety," *Proc. Natl. Acad. Sci. USA*, vol. 98, no. 1, pp. 688-693, 2001.

(Sloman et al., 2005) A. Sloman, R. Chrisley, and M. Scheutz, "The architectural basis of affective states and processes," in *Who Needs Emotions: The Brain Meets the Robot*, J. Fellous and M. Arbib, Eds. New York, NY: Oxford University Press, 2005, pp. 203-244.

(Snyder and White, 1982) M. Snyder and P. White, "Moods and memories: Elation, depression, and the remembering of the events of one's life," *Journal of Personality*, vol. 50, no. 1, pp. 142-167, 1982.

(Strack et al., 1988) F. Strack, L.L. Martin, and S. Stepper, "Inhibiting and facilitating conditions of the human smile: A nonobtrusive test of the facial feedback hypothesis," *Journal of Personality and Social Psychology*, vol. 54, no. 1, pp. 768-777, 1988.

(Sugeno, 1985) M. Sugeno, *Industrial Applications of Fuzzy Control*, New York, NY: Elsevier Science, 1985.

(Sutton and Barto, 1998) R. Sutton and A. Barto. *Reinforcement Learning*. Cambridge, MA: MIT Press, 1998.

(Thrun, 2003) S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millennium*, G. Lakemeyer and B. Nebel, Eds. San Francisco, CA: Morgan Kaufmann, 2003, pp. 1-35.

(Tingley and Browne, 2006) C. Tingley and W.N. Browne, "Developing an emotion-based architecture for autonomous agents," *Proc. 3rd International Conference on Autonomous Robots and Agents*, pp. 225-230, Palmerston North, NZ, 2006.

(Tomkins, 1984) S.S. Tomkins, "Affect theory," in *Approaches to Emotion*. K.R. Scherer and P. Ekman, Eds. Hillsdale, NJ: Erlbaum, 1984, pp. 163-195.

(Ulrich and Borenstein, 1998) I. Ulrich and J. Borenstein, "VFH+: Reliable obstacle avoidance for fast mobile robots," *Proc. IEEE International Conference on Robotics and Automation*, pp. 1572-1577, Lueven, Belgium, 1998.

(Ulrich and Borenstein, 2000) I. Ulrich and J. Borenstein, "VFH*: Local obstacle avoidance with look-ahead verification," *Proc. IEEE International Conference on Robotics and Automation*, pp. 2505-2511, San Francisco, CA, 2000.

(Velásquez and Maes, 1997) J.D. Velásquez and P. Maes, “Cathexis: A computational model of emotions,” *Proc. 1st International Conference on Autonomous Agents*, pp. 518-519, Los Angeles, CA, 1997.

(Weiner and Graham, 1984) B. Weiner and S. Graham, “An attributional approach to emotional development,” in *Emotions, Cognition, and Behavior*, C.E. Izard, J. Kagan, and R.B. Zajonc, Eds. New York, NY: Cambridge University Press, 1984, pp. 167-191.

(Wierzbicka, 1992) A. Wierzbicka, “Talking about emotions: Semantics, culture, and cognition,” in *Cognition and Emotion*, vol. 6, no. 3/4, pp. 285-319, 1992.

Appendix A: Terms and Abbreviations

Adaptive control – A control approach whose parameters can be adjusted dynamically in response to changing conditions.

Affect – An external manifestation of emotion, mood or related state.

Cognition – High-level mental functions such as awareness, perception, memory and learning.

DAQ – Data Acquisition.

Deliberative control – A control approach that employs global world models to plan future actions.

Drive – An affective state corresponding to a need or desire that motivates a behavioural response.

Emotion – A short-term affective state that possesses intentionality towards specific conceptual or real objects.

Exteroceptor – A sensor that measures an aspect of the environment external to the robot.

Feeling – A physiological response to affect, such as a change in heart-rate, skin conductivity and temperature.

GPS – Global Positioning System.

Interoceptor – A sensor that measures an internal property without requiring feedback from the external environment.

IR – Infrared.

Mood – A long-term, non-specific affective state.

Personality – A permanent (or very long term) affective characteristic.

Reactive control – A control approach that favours local sensor data and fast response to environmental dynamics.

SLAM – Simultaneous Localisation And Mapping.

Stimulus – An event that elicits a behavioural response.

Strong AI – A machine possessing the subjective aspects of consciousness.

TD – Temporal Difference (learning).

Valence – Intrinsic attractive (positive) or aversive (negative) property of an object, event or situation.

VFH – Vector Field Histogram.

Weak AI – A computational model inspired by theories of biological intelligence.

Appendix B: DVD Contents

The DVD attached to this thesis contains the following:

- Electronic copy of thesis
    - Microsoft Word format
    - PDF format
    - Figures
- Experiments
    - Environment data
    - Experimental data
    - Video samples
- Source code
- Datasheets