Autonomous Learning of Appropriate Social Distance by a Mobile Robot

Yang Wang

A thesis submitted in partial fulfilment
of the requirements of the University of Hertfordshire
for the degree of Doctor of Philosophy

School of Electronic, Communication and Electrical Engineering

University of Hertfordshire

June 2008

Abstract

This thesis aims to design an appropriate human-following solution for a mobile robot. The research can be characterised as interactive model building for a Human Robot Interaction (HRI) scenario. It studies possible proposals for the robot system that learns to accomplish the task autonomously, based on the human preference about the positions and movements of the robot during the interaction. A multilayered feedforward network framework with backpropagation is the adopted learning strategy.

The research breaks the task of following a human into three independent behaviours: social positioning, human avoidance and obstacle avoidance. Social positioning is the behaviour that moves the robot, via reasonable paths, to the most appropriate location to follow the human. Both the location and the paths reflect the preference of the human, which varies by individual. The main body of the research therefore proposes a using-while-learning system for this behaviour such that the robot can adapt to the human's preference autonomously.

This research investigated multilayered feedforward networks with backpropagation learning to fulfil the social learning task. This learning model is less used in HRI because a complete set of correct training data doesn't exist as the human preference is initially unknown. The research proposes a novel method to generate the training data during the operation of learning and introduces the concept of adaptive and reactive learning. A novel training scheme that combines the two learning threads has been proposed, in which the learning is fast, robust and able to adapt to new features of the human preference online. The system enables the behaviour to be a real using-while-learning system as no pre-training of any form is needed to ensure the successful performance of the behaviour. Extensive simulations and interactive experiments with humans have also been conducted to prove the robustness of the system.

Acknowledgements

First of all, I have to pay my greatest gratitude to my parents who have been supporting me unconditionally for all these years. I also would like to thank my family and relatives who have helped me so much. Without them, this thesis would have never happened.

I am very grateful to my supervisor Dr. Lee, who has been very warm and patient to me during my research. I have learnt far more than I expected from him and his contribution to this thesis has been greatly appreciated.

I have got great friends to help me through these four years, particularly, Jon, Forest, Chinghong, Bin and his lovely wife Tingfang. My life would have been a lot more difficult without them. Also, I would like to thank Howard and Dirk for their help on the proofreading.

For others that I do or do not remember, to whom I owe so many thanks, I would like to present my appreciation with the greatest gratitude.

Contents

List of Figures

Terminology

Adaptive learning	The backpropagation learning that contributes to the overall generalisation and convergence.
Flat surface	The area with a gradient of zero in the human reward surface.
Lower peak	This specifically refers to one of the reward peaks, which is geographically located in the lower half of the reward surface. It is used in Chapter 6 only.
Reactive learning	The backpropagation learning that concentrates on the fast learning of local situations and environment.
Reference frame, global	The frame of reference where the stationary objects in the environment are still. Often uses the initial robot position as the origin of its coordinates.
Reference frame, human	The frame of reference where the human is always still and at the origin of its coordinates facing right.
Reference frame, robot	The frame of reference where the robot is always still and at the origin of its coordinates facing right.
Reward peak	The area in the reward surface that has the highest reward value.
Reward score	The value of feedback given by human, based on the robot movements. This is also the value of the human reward surface at each position.
Reward slope	The area of reward surface that has non-zero gradient.
Reward steps	The total number of levels of values that appear in a reward surface.
Reward surface	The function that represents the human preference, which is used to rank the performance of the robot movements.
Session	A set of walks (in simulation), in which the robot

	tries to position itself socially.
Walk	A set of movements from the start point of the robot to the end of the movements where either the goal is reached or simulation/interaction ends or expires.
Upper peak	This specifically refers to one of the reward peaks, which is geographically located in the upper half of the reward surface. It is used in Chapter 6 only.

Mathematical Notation Convention

Space

Spaces are denoted by uppercase roman letters with Euclid Math font. For example:

$\mathcal{A}$: general action space of a system;

$\mathcal{X}$: general state space of a system;

$\mathcal{Y}$: general output space of a system;

$\mathcal{P}$: the robot position space (social behaviour state space).

Vector and Matrix

Vectors are denoted by bold lowercase letters and matrices are denoted by bold uppercase letters. For example:

p: the robot position vector in the human frame of reference;

q: the human position vector in the robot frame of reference;

x: general state vector;

s: displacement;

W: the neural network weight matrix.

Limitation Constant

The limitation constants are denoted by italic capitals. For example:

K: the limit of the speed;
H: the limit of the learning rate;
D: the neighbour learning distance limit.

Learning Rate

Greek letter η is used specifically to denote learning rate in this thesis. For example:

η : general learning rate notation;

η_r : the reactive learning rate;

η_s : the scalar for the secondary scoring surface.

Time Index

Superscript, *k* and *t* are used specifically to denote the time index where there is no confusion with power. For example, $\mathbf{p}^k$ is the robot position vector at time index *k*;

Quantised/discrete Elements

The circumflex (^) over the top of an element means this element is either quantised or discrete. For example:

$\hat{\mathcal{A}}$: quantised robot action space;

$\hat{\mathcal{P}}$: quantised robot position space;

$\hat{a}$: quantised robot action state;

$\hat{\mathbf{p}}$: quantised robot position vector.

System Function

Greek letter μ is used specifically to denote the system function in this thesis. For example:

μ : desired system function;

$\hat{\mu}$: the discrete estimation of the desired system function;

$\tilde{\mu}$: the continuous estimation of the desired system function.

Natural Logarithm

Italic roman letter *e* is used specifically to denote the natural logarithm constant.

Vectors Observed in Global Frame of Reference

Prime is used to denote vectors observed in global frame of reference exclusively in Chapter 7. For example:

p': the robot position vector observed in the global frame of reference;

s': the displacement vector observed in the global frame of reference.

Vectors Observed in Different Time

The vertical bar, |, with the subscription of the time index at the right side denotes the frame of reference in which current vector is observed, which is used exclusively in Chapter 7. For example, $\mathbf{p}^{u}|_{k}$: the observation of vector **p** from the coordinate frame at time *k* of the robot position vector measured at time *u*.

System Reward Functions

R is used to denote the reward surface in this thesis, which is presented as a function of position, ***r***(.). R_s denotes the secondary scoring surface.

Chapter 1 Introduction

This thesis aims to build a mobile robot behaviour that can maintain the appropriate social position autonomously while following a human. In the proposed research scenario, the human uses a performance index feedback to express their level of satisfaction about the robot's performance, but neither the human preferences about social distance nor the desired positioning policy of the robot is known initially. This thesis addresses the problem of constructing a using-while-learning behaviour that learns to position the robot appropriately online, based on feedback from the human.

This research is motivated by two rapidly growing fields in robotics: Human Robot Interaction (HRI) and autonomous machine learning. The aim of HRI research is to develop robot systems and behaviours that fulfil some tasks in a social environment. The robot is required to behave like a social member to perform a certain social function. In the wide range of applications seen in recent years, besides the challenges introduced by conventional robotics, a central debate in HRI concerns the ways to model robot behaviours to match the human's social preferences. This unavoidably involves modelling human behaviour. The use of common solid social models from psychology has become one of the most commonly used methods for researchers. Many other researchers have started with HRI experiments in order to build a trustworthy interactive model based on the samples of human responses towards robots before modelling the autonomous robot behaviour. In either case, the model of human preference plays a significant role in how the robot behaviour is modelled. The fitness of the robot behaviour is largely subject to the correctness of the predefined human behaviour model that is assumed to be true universally among all human beings that the robot may encounter. It is, however, obvious that variation is one of the most important features of human behaviour. Acting as a social member, the robot has to have the ability to respond to the dynamics of other social members and to respect individual differences among humans. This thesis therefore seeks to develop a robot behaviour that dynamically adjusts itself to the human preference, and begins with a minimal model of what that preference might be.

Autonomous machine learning has provided an attractive option for this adaptive robot behaviour, especially backpropagation learning with its ability to generalise. Traditionally, autonomous learning is developed for estimating a function where some examples of the function are known and can act as the teacher during the learning. The HRI scenario here has presented a new challenge because the human preference, which is essentially the function that must be learnt, is totally unknown initially and useful information can only be collected through the human's feedback while the robot is operating. The learning thus has to take place while the system is working. This is known as online learning. However, online learning applications in HRI still largely remain unexplored. Because of the engagement of the human, the prior knowledge that can be offered to the learning system is little, the possible uncertainties involved in the learning are significant and the useful information, i.e. the human's feedback, is limited. The learning system has to learn as soon as the robot starts operating and has to learn fast enough so that the human suffers a minimum amount of dissatisfaction. The development of such a learning system is another challenge faced by this thesis.

The novelties of this thesis come from both HRI and autonomous. From the overlaps of the two fields, the major goal of the research emerges, which is to achieve an autonomous using-while-learning socially interactive robot behaviour. In this Chapter, Section 1.1 will first summarise our contributions to knowledge and Section 1.2 will provide an overview of the following chapters of this thesis.

1.1 Contributions to Knowledge

The central aim of this thesis is to propose an autonomous using-while-learning system for an interactive behaviour in an HRI scenario: following a human. The research has been presented in two international conferences (Wang and Lee, 2006a, 2006b). The main contributions of this thesis can be summarised as:

1. The design of a successful using-while-learning HRI behaviour without any pre-training, which gives the social behaviour the ability to generalise, the ability to work with little prior knowledge of human's preference, fast adaptation to the individual preference as well as flexible adaptation to the new features of the human preference discovered during interactions;
2. A novel online training data collection method that solves the problem of the lack of training data for multilayered feedforward neural networks in the HRI

scenario, which enables multilayered feedforward neural networks to be used in the social interactive robot behaviour;

3. The novel proposal of using two learning threads with different learning speeds and targets. This gives the system fast learning speed, good convergence and the flexibility of learning new features quickly while maintaining existing knowledge, which enables the system to be truly using-while-learning in interactions engaging human behaviour;
4. A new assessment routine tailored for the proposed using-while-learning system that employs not only error performance but also a set of system measurements including the system outputs, reward history and the training data set behaviour to monitor and analyse the system's online behaviour. A thorough understanding of the behaviour of the system and the two learning threads during the operation of the system is established through a cross-comparison of these measurements;
5. A new method to give the conventional reinforcement learning the ability to generalise by introducing a secondary reward surface. This surface smoothes the system reward surface so that the states that originally have no trainable information can learn effectively as well;
6. A learning speed enhancement of reinforcement learning by adopting the idea of neighbour learning from competitive networks. By assuming that nearby states in the system should have similar actions, this method significantly increases the learning speed of conventional reinforcement learning.

1.2 Overview of the thesis

The remainder of the thesis is structured as follows:

Chapter 2 introduces the context of this research. This thesis is built upon the fields of HRI, autonomous learning and behaviour-based robotics, the concepts of which are reviewed in this chapter. It presents the proposed human-following scenario and the core interactive behaviour, social positioning, which the robot needs in order to fulfil the human-following task. This chapter identifies the directions of the research, which the rest of the thesis follows.

Chapter 3 studies the learning model for the interactive behaviour. The model is an abstract from existing research on the understanding of human preference in the context of our interactive scenario. This chapter describes the basic configuration of

the learning system as well as the learning objectives, on which the research into learning systems is based.

Chapter 4 studies the use of reinforcement learning in the proposed model. The study modifies the reinforcement learning algorithm to fit the needs of using-while-learning. Analysis of the simulation results reveals both the strength and the weakness of this popular form of learning, which shows the necessity for a study on alternative learning methods in social positioning behaviour.

Chapter 5 proposes to use multilayered feedforward neural networks for the learning task so that the system can benefit from their ability to generalise. This chapter first introduces an online training data collection method to overcome the problem of lack of training data. Then the study presents simulations in a similar setting to the one used for reinforcement learning in Chapter 4. The comparison of the results supports our choice of the new learning method. Concluding that the proposed leaning method is successful, this chapter raises the further question of the system's fitness in a more complex situation.

Chapter 6 identifies the weakness of the proposed neural learning in social positioning, in that it is unable to adapt to future changes in the environment or the human preference. To solve this problem, this chapter first introduces the concept of adaptive and reactive learning threads. It then proposes the novel training routine of training the system with both learning threads simultaneously. The analysis of the successful simulations not only provides concrete support for the proposal but also presents thorough understanding of the effects of the two learning threads during the operation. Placing the system in a more complex situation, some interactive experiments with human operators then are presented at the end of this chapter.

Chapter 7 discusses the robot control system in order to bring the proposed scenario from simulation to reality. This chapter studies the general control structure of the robot system and the necessary sensor for human position measurement. The kinematics of the robot are also analysed and the chapter concludes with experiments with the robot and a learnt social positioning behaviour.

Chapter 8 discusses our major findings in this thesis, which is the using-while-learning framework consisting of the online training data selection and the neural network with two learning threads. After summarising our contributions to knowledge, this chapter extends our finding into other possible applications and discusses the advantages that could be achieved with our framework, compared to

existing results in those applications areas. Finally, we discuss possible future research directions and improvements in our research.

Chapter 2 The Context of the Research

The aim of the proposed project in this thesis is to build an autonomous learning behaviour for a mobile robot to follow the human at an appropriate social distance. The proposed scenario, following a human, concerns the social interaction between the human and the robot which belongs to one of the popular current research areas: HRI (Breazeal, 2003, 2004). The targeted problem solving method, an autonomous learning system, has shown a rising potential in adaptive control systems and artificial intelligence. Also, being an application of robotics, the concept of behaviour-based systems (Maes, 1993) that has been widely adopted in contemporary robotics is also a foundation of our research. As a project concerning such wide areas, the perspectives and methodologies existing in the literature are plenty. The objective of this chapter is therefore to position this thesis in the related literature, establish the interests of our research, and identify the key challenges and tasks.

In the remainder of Chapter 2, Section 2.1 reviews the background of the human following scenario and HRI. Section 2.2 looks into the existing methods of autonomous learning in social robotics. Section 2.3 analyses the human-following task based on behaviour-based concepts and identifies the core interactive behaviour for accomplishing the human-following task: the social positioning behaviour. Finally, Section 2.4 establishes the context of the research by stating the research objectives.

2.1 Human Following: An HRI Scenario

The emergence of the concept of social robots was an outcome of the research into animates, where the early social behaviours studied were ant and swam robots, the so-called *collective robot behaviour* (e.g. Deneubourg et al., 2000, Kube and Bonabeau, 2000, Melhuish et al., 1998, Stricklin et al., 1995). Bonabeau (1999) points out that although those robots appear to work separately, they produce complex behaviour patterns collectively. Given the growing interest in developing robots situated in human society, researchers have been looking into social robots that produce social patterns or finish social tasks individually. Dautenhahn (1997) argues

that for those robots, they act as members of society and have characteristics that adhere to societal norms and conventions. Dautenhahn and Billard (1999) define such individualised social robots as follows:

> 'Social robots are embodied agents that are part of a heterogeneous group: a society of robots or humans. They are able to recognise each other and engage in social interactions, they possess histories (perceive and interpret the world in terms of their own experience), and they explicitly communicate with and learn from each other.'

Social interaction has been recognised as one of the most important features of a social agent by social scientists such as Ashworth (1979) and Cairns (1979). Efforts made by robotics researchers to study the issues of social interactions between robots and humans are enormous. Breazeal (2003) defines classes of social robots in terms of the complexity of social interactions, Scassellati (2000) and Tews et al. (2002) identify some fundamental issues for humanoid robots, and Dautenhahn (2004) studies the issues of a robot as a life companion. While researchers are trying to introduce social robots into human society, how to model a successful interaction between the robot and the human has become a widely studied area. This is the discipline that looks into the design of the robot behaviours so that the robot can interact with a human successfully, where the accomplishment of such an interaction is always associated with the personality or preference of the human. This field of research is known as Human Robot Interaction, of which Fong et al. (2003) provide a useful survey.

2.1.1 Human Robot Interaction

In the field of HRI, one of the fastest growing areas is peer-to-peer interaction (e.g. Murphy, 2004, Skubic et al., 2004, Triesch et al., 2006, Nakauchi, 2002). That is the interaction between one robot and one human. The interest in this field is essentially to investigate possible robot designs to satisfy certain social functions in human society. Many researchers have identified practical motives and initiatives for HRI development. Dautenhahn, for example, has presented possible domains and desirable social skills for social robots (Dautenhahn, 2003), as well as their design issues (Dautenhahn, 1999). Paulos and Canny (2001) on the other hand presented the social robot as a representation or representative for the human. Research in HRI is

growing very fast at present but the practical future of the research still remains vague and the possible roles for social robots to play in the human world as identified by these researchers are still distant goals (Dautenhahn, 1997, Dautenhahn and Billard, 1999). Therefore huge potential remains in this area and plenty is yet to be revealed.

Because the social interactions are so complex in the human world, HRI projects in the literature have divided the interactions into many different aspects, each of which has attracted a large number of researchers. The interactions are normally divided by the means of communication. Facial expression (e.g. Breazeal, 2000, Canamero and Fredslund, 2001, Kobayashi et al., 1994, Scheeff et al., 2000) is a popular example of those interactions being pursued by many researchers. Body language has also been widely studied (e.g. Skubic et al., 2004, Kanda et al., 2002, Song-Yee et al., 2000). Imitation also generates huge interest (e.g. Schaal, 1999, Bakker and Kuniyoshi, 1996, Hirzinger, 1996).

The robot behaviour that concerns only the robot position has already attracted lots of researchers in non-interactive scenarios, typically obstacle avoidance (e.g. Laue and Röfer, 2005, Zavlangas et al., 2000) and localisation (e.g. Fox et al., 1999, Jetto et al., 1997, Demirli and TRurksen, 2000, Tardós and Neira, 2002). It is, however, not until recently that the issue of the positioning of robot in the presence of a human has become another huge interest of research. Nicolescu and Mataric (2001) present their research concerning a robot that learns a positioning strategy from human's demonstration. Yoda and Shiota (1996) propose to pass and avoid collision with a human while avoiding any human discomfort. Nakauchi (2002) studies how a robot stands in line with humans. How to position the robot appropriately to avoid the discomfort of a human is the question that these researchers aim to answer. In fact, it has been long established in psychology and social science that social agents are sensitive to the spaces around them (e.g. Malmberg, 1980, Sack, 1986, Secord and Backman, 1964). Bringing their definition of such problems forward into HRI research, the controlling of the robot position as a means of interaction with human is known as social distance interaction.

2.1.2 Social Distance Interaction

Hall (1966) points out that social space is an important element of any form of social organisation and that the control of distance between agents is an important means of communication. The humans' preference about their social space defines a desired

manner of the positioning of other social agents around them. In HRI research, modelling human behaviour and preferences plays an important role. How the human preference is modelled directly regulates the outcome of the robot behaviour. It is known that humans react differently according to the position of other agents around them. Secord (1964) points out that their reactions commonly indicated certain regularities of a person's feelings, thoughts and predispositions to act towards some aspect of their environment. Many HRI research projects have adopted models from psychology to build a model of a human's preference about social space in certain scenarios. They are mostly models of the human's comfort level over the space around them. The most commonly used model of this type describes the 'passing a human' scenario, and has been well studied in previous social science research. Figure 2-1 is the model of the human's preference of social distance while someone is passing them, used by Pacchierotti (2006). This surface with ellipse-shaped cross sections has also been used in many other HRI applications involving passing a human (e.g Chen et al., 2004, Yoda and Shiota, 1997, Simmons et al., 1997).

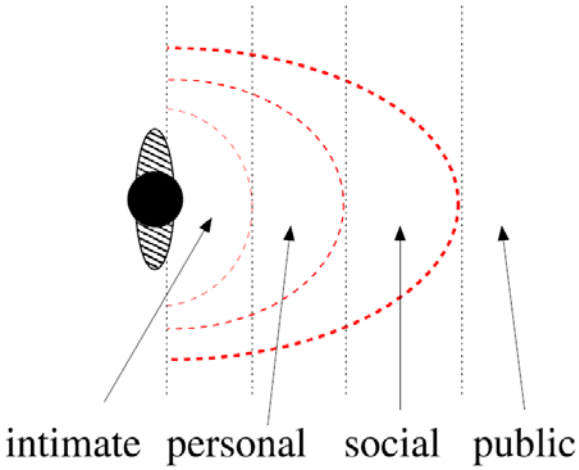


Figure 2-1 The human preference of spatial distance zones in the scenario of a robot passing a human. It generally describes how the human feels about an agent regarding to the distance and position between them (Pacchierotti et al., 2006)

Based on this human preference, Nakauchi (2002) discovered that, in the case of a robot trying to stand in line with humans, their preference turns out to be a diamond-shaped surface, as shown in Figure 2-2. He achieves this result based on the calculation derived from the human preference in Figure 2-1 and bases the rule of stand-in-line behaviour on such a human preference model.

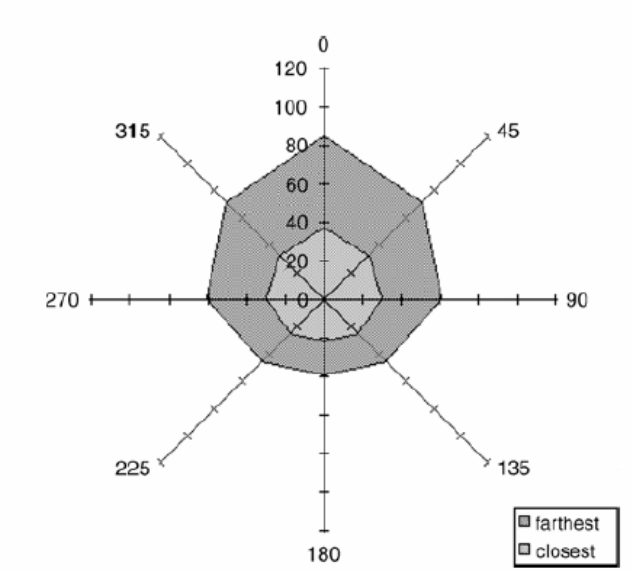


Figure 2-2 The human preference of spatial distance zones when a robot tries to stand in line with humans. (Nakauchi, 2002)

The HRI scenario proposed by this thesis is following a person. It is another one of the most common scenarios falling into the context of interaction over social distance but the model of the human preference has not been well defined. For a social behaviour in which no previous human preference model can apply, researchers normally conduct experiments with humans to draw out the model. For example, in the research into a home companion robot, Dautenhahn et al. (2006) collected the data from experiments of how a human would like to be approached by a robot. They conclude a robot movement policy based on the human preference as shown in Figure 2-3. In Figure 2-3, the dots are values assigned to positions. They are costs of a robot being in those positions. The task of the robot is to approach the human with lowest possible cost, i.e. avoiding the highly valued position on the map.

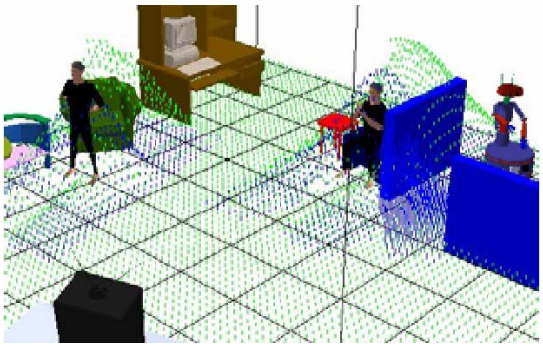


Figure 2-3 The cost of robot action based on the human preference in home companion robots. The higher the value on the map, the higher cost for the robot to take action, i.e. the human is less satisfied (Dautenhahn et al., 2006)

In their research into the human's preference of social distance, the methodology is to measure the human's feedback with different robot behaviours. Such feedback reflects the human's attitude that is essentially the requirement that the robot is trying to satisfy. In current HRI research, the most commonly used method is to ask the human in the interaction to indicate their feelings through a handheld device. Koay et al., (2005) studied the issues involving using a handheld feedback device in HRI, a device that has been used in the research carried out by Dautenhahn et al. (2006) and Woods et al., (2006). Similar device has been used in various social robot applications (e.g. Calinon and Billard, 2006, Syrdal et al., 2006, Blumberg et al., 2002). By using such measurable feedback, a direct appraisal of the robot performance in the social context is available. Researchers thus can sketch the model of the human preference tailored to the HRI scenario they propose.

The availability of measurable human feedback gives the research a new possibility of modelling the robot behaviour without fully known human preference model because such feedback is essentially a partial presentation of the human preference. Kanda (2003) uses only some basic rules of human interaction but leaves the exact robot behaviour to be adjusted by the human feedback, which involves the adjustment of social distance. A similar technique has been adopted by Kaplan (2002) using a clicker feedback device to control a simulated robot behaviour. With the human feedback, the requirement of human preference can be come less strict as it is possible to abstract the human model from their feedback. An attractive idea thus is to have a robot behaviour that needs no previous knowledge of the human preference but can adapt itself to the human preference based on the feedback during operation and produce reasonable interaction behaviour. This can offer highly flexible robot behaviours in a dynamic social environment. It can be particularly useful for a behaviour such as human following because the human preference of being followed can vary a lot among individuals and can change in different environments, which makes it very difficult to model the human preference beforehand. It is easy to understand that in a narrow crowded place, the human would like the follower in a different place from their preference in a wide quiet environment.

It is reasonable to believe that human preferences are distinguished from one to another and that the individual uniqueness can play an important part in the social interaction. Given the unpredictable elements in the nature of the interaction in human following, a dynamic adaptive system approach is promising. This thesis aims to build

a system that learns how to position the robot based on the human preference presented by their feedback during the interaction so that the robot's behaviour respects and adapts to the human's individual preference. The availability of the measurable human feedback gives the possibility to dynamically form the social positioning method of the robot in HRI. One of the adaptive system solutions which have been given a large amount of attention is autonomous learning.

2.2 Autonomous Machine Learning

Autonomous machine learning has been widely used to construct dynamic control systems that are highly adaptive to the environment. Such use of machine learning is mostly to estimate a target function based on some limited examples of that function. Those examples act as the teacher of the learning system and this type of learning is called *supervised learning*. The learning is autonomous because the system adjusts itself based on certain rules and the examples presented to the system. This self-adjustment process of the learning system is known as *training*. Many autonomous learning systems exist in the literature and remain highly active research topics.

Learning systems have been introduced into many HRI applications. The human preference can be understood as a map that directs the robot movements. Some research refers to such a function as *human decision state* (Nicolescu et al., 2006). In the research into learning from demonstration (e.g. Schaal, 1999, Atkeson and Schaal, 1997), they propose a system that acquires skills to finish difficult tasks by mapping such human decision functions through learning examples presented by humans. Similar concepts have also been adopted in human-robot skill transfer (e.g. Grudic and Lawrence, 1996, Asada and Liu, 1991). These are some of the examples where the research is not based on accurate pre-calculations of the human preference but on adaptive methods to build robot behaviours by learning from the human. Among various published machine learning studies, two which especially interest us in our research are: reinforcement learning (Kaelbling et al., 1996) and neural networks (Patterson, 1996).

2.2.1 Reinforcement Learning

Reinforcement learning was initially proposed based on mathematical psychology (Hilgard and Bower, 1975). It has become a very popular algorithm since then. Reinforcement learning is essentially a discrete map estimator. For any system whose behaviour can be described as a set of possible actions over a set of possible states, knowing some kind of performance index of the system actions, reinforcement learning is able to create a map between quantised system states and actions to estimate their real desired relationship. The estimating, i.e. the training, takes place by counting the likelihood of executing each quantised action at each quantised state based on the performance appraisal of the system. Different rules for this counting of the likelihood define different types of reinforcement learning. One of the early proposals is the *linear reward-inaction* rule, the convergence of which has been proved by Narendra (1974). More recently, *Q-Learning* proposed by Watkins (1989) has established its place as the most commonly used reinforcement learning rule. Chapter 4 gives more detail of the reinforcement learning algorithm.

The use of reinforcement learning in robotics has been a very successful research field, typically the research into dynamic programming as opposed to prior programming of robot behaviours. The initiative is to build a successful robot behaviour that is difficult to model directly from physical and mathematical models. This approach has been applied to learning from demonstration (Schaal, 1997) and some HRI applications (e.g. Nicolescu et al., 2006, Kasper et al., 2001, Atkeson and Schaal, 1997, Wang and Lee 2006a).

Reinforcement learning has a great generality. It quantises the system into a set of states. By assigning each action at each state a reward, the learning can find the desired transition between states and the corresponding actions in a quantised map. Reinforcement learning has been one of the most studied learning methods and has appeared to have dominance in the learning of interactive robot behaviour. Compared to conventional control methods, reinforcement learning functions without requiring the desired output examples of the system, in other words the system error performance is not needed. The learning is based on a reward that indexes the performance of the system, which is a much easier condition to meet than finding the desired system output or system error measurements in many HRI applications. The desired performance of a robot in HRI is always defined by the preference of a human

and is normally unknown. But certain forms of ranking index on the robot performance are, however, normally achievable. This index can be constructed either from perception of human as mentioned in Section 2.1.3 or from some mathematical rules. Both are much more reliable or easier to get than the estimated examples or models of the human preference (e.g. Mitsunaga et al., 2005, Isbell et al., 2001). This then explains the great attention paid to reinforcement learning by the academics in the learning of HRI. As a matter of fact, such popularity of reinforcement learning is not only in HRI but across most areas of robotics.

2.2.2 Neural Networks

Another alternative form of autonomous learning that has been growing very fast is *artificial neural networks*. Although, the use of neural networks in robotics in existing literature is still largely limited to conventional control problems (e.g. Fierro and Lewis, 1998, Lewis et al., 1995), and rarely to the learning of a structured social behaviour, we have a particular interest in this learning because of one of its distinctive advantages over reinforcement learning: the ability to generalise, which gives the neural network the ability to perform in an unlearnt situation based on learnt knowledge.

The foundation of modern neural network learning is based on Hecht-Nielsen's (1989) proposal of error backpropagation learning, in which he defines a neural network as follows:

> A neural network is a parallel, distributed information processing structure consisting of processing elements, the *neurons*, interconnected together with unidirectional signal channels called *connections*, each of which carries a single discounting factor called *weight*. Each neuron has a single output which branches into as many collateral connections as desired. The output function of a neuron can be of any mathematical type desired. All of the processing that goes on within each neuron must be completely local. That is, the output of a neuron must depend only upon the current values of the input signals arriving at the processing neuron via impinging connections and upon values stored in the processing neuron's local memory, which are the backpropagation factors from previous layers.

Supervised neural networks are more general function estimators than reinforcement learning because the map established by neural networks is continuous. Neural networks have been greatly used in control systems (e.g. Ahmed et al., 1995, Bryson and Ho, 1969, Chen, 1990, Zhao et al., 2002) and function estimating (e.g. Battiti, 1992, Foresee and Hagan, 1997, Werbos, 1988). They generate the system function as a continuous map between the input and output space. Therefore, when the learning is finished, the system can provide reasonable outputs for any input value from the defined input space, no matter whether those input values have been exposed to the system or not, hence the ability to generalise. This appears to be a huge attraction to us because the ability to conduct reasonable behaviour in unknown situations based on current knowledge will be an advantage in any HRI scenario. However, as we can see, the learning of neural networks is based on some known examples of the desired outputs of the system. This has become an obstacle in applying neural networks in the social learning of robots as the desired output of a social robot behaviour is normally defined by the human mind and it is difficult to model desired system outputs. A performance index given by the human as a feedback is what most researchers can achieve in many applications, as mentioned above, and it is the human feedback that we have in our research.

It is clear that both types of autonomous learning discussed above have their attractions to fit into the HRI scenario. Reinforcement learning can learn without knowing the desired system outputs but only from the human feedback. Neural networks have difficulty to learn with human feedback but can generalise based on known knowledge. It is unwise to decide which approach to adopt before any concrete investigations have been done, which are located in Chapter 4 and 5. However, whichever method that might be chosen, it has to have the ability to perform the learning online because the human feedback is not known beforehand and training before use is not possible. Moreover, for a realistic robot behaviour, the system has to behave reasonably even when it is still learning. This brings up another important feature of our robot behaviour: the ability of using-while-learning.

2.2.3 Using-while-learning

Autonomous learning methods can be applied to a system in two ways: offline and online. Traditionally, a system is trained before use. The introduction of learning was aimed at, and is still widely used in, problems that are difficult or impossible to model

but for which some information, such as some examples of the input and desired output of the system, is known. The use of the learning network is then to train the system fully so that the system has a close enough presentation of the desired system function and it is put into operation afterwards. This process of training before operating is now known as offline learning or pre-training.

With the growing interest in adaptive systems, a big challenge that researchers face is systems with factors that are unknown or unpredictable at the design stage. In learning systems, many problems that are difficult to model have no trainable data beforehand. Human-following behaviour is one such problem as the model of an individual preference can neither be modelled nor predicted by some prior examples. Pre-training before the operation of the system therefore can't be organised. For systems like this, researchers propose to train the system while the system is operating. The learning then adjusts the system performance while the system is being used. This kind of learning system is known as an online learning system. One of the most typical uses of an online learning system is goal finding. In order to find an unknown goal, the learning is performed while the system is operating. As the learning adjusts the system dynamically, the goal is achieved at the end of the operation.

Being an online learning system, it is not trained before use and has to rely on random outputs before the system learns. However, for some systems, long term random outputs may not be practical. For example, in the human-following robot behaviour, long time random movements of the robot are neither practical for the motor system nor acceptable for the human. An online learning system is not always usable online. This is the reason that, in this thesis, we will use the term "using-while-learning" to describe the desired ability of online learning in the proposed systems. We use this term to emphasise that, in order to behave successfully in the HRI scenario, the system needs not only to learn while operating but also to produce usable outputs while learning. Therefore the goal of the learning system that this thesis aims to achieve is a using-while-learning system that learns based on the online feedback of the human in order to follow the human appropriately.

Both reinforcement learning and neural networks have been known to be able to work online. In this thesis, we will investigate both the learning behaviour of reinforcement learning and the learning of interactive robot behaviour by neural

networks before we select the more appropriate method based on their learning performance and their using-while-learning ability.

2.3 Behaviour-based Structure

Behaviour-based structures (Arkin, 1998, Maes, 1993) have now been widely adopted in robotics. In contrast to traditional planner-based robot applications (e.g. Beetz et al., 2001, Ge and Cui, 2000), behaviour-based structures split a complete task into several simple behaviours and it is argued that by completing each simple behaviour and combining their outputs, complex tasks can be achieved. Contemporary research projects in robotics are mostly based on behaviour-based structures.

A typical case of this is the split between the goal-finding behaviour and collision avoidance behaviour. We identify the behaviours involving completing the human following task as follows.

Obstacle avoidance:

Obstacle avoidance is a typical behaviour for avoiding collisions, which acts independently in behaviour-based robot systems. It is one of the most well studied robot behaviours, and rich research literature exists. There are many well studied methods, most of which are based on range sensors or cameras. A typical early application is obstacle avoidance using potential fields (e.g. Ge and Cui, 2000, Koren and Borenstein, 1991). Fuzzy logic has been widely introduced in obstacle avoidance as well (Hoffmann, 2004, Jetto et al., 1997, Demirli and TRurksen, 2000). A wide range of variations have been presented for this behaviour (e.g. Laue and Röfer, 2005, Zavlangas et al., 2000, Tardós and Neira, 2002).

Human avoidance:

Although obstacle avoidance is the behaviour that is most often referred to for dealing with collisions, HRI researchers have suggested that the avoidance of a human is a different behaviour from avoiding an object (e.g. Yoda and Shiota, 1997, Pacchierotti et al., 2006). Passing a human, as has been reviewed in Section 2.2, is one such behaviour. It is listed differently from traditional obstacle avoidance because, while avoiding the human, minimising the human's discomfort is always a requirement for social robots. Thus the human preferences play an important role in

this behaviour. The typical human preference model applied is the model used in passing human behaviour as shown in Figure 2-1. This preference model has been widely accepted by researchers.

Social positioning:

Social positioning is the behaviour in which the robot interacts with human through the human feedback and adjusts its position and movement. It is the 'goal-finding' behaviour and the main interactive component of the human following task. The positioning of the robot is solely defined by the human preference. Existing studies in HRI haven't addressed any possible human preference that is suitable universally. Unlike human avoidance behaviour, the personal preference among humans in the context of being followed varies among individuals. Even the preference of the same person can change due to the environment. Social positioning behaviour remains less studied in literature and is, therefore, the major target of this thesis.

By separating social positioning behaviour from collision avoidance behaviours, it is reasonable to exclude possible obstacle avoidance from the social positioning model. The behaviour-based system is based on the cooperation of different behaviours, each of which completes a separate task in an independent model. By assigning collision avoidance to separate behaviours, we can study the social positioning behaviour in a collision-free model. The collision-free assumption here includes both the avoidance with objects and the collision between the human and the robot.

The perception of the robot is measured relative to the human. Therefore, the control of the robot by the social positioning behaviour is in the *human reference frame* that offers the coordinate. In this frame, human is placed at the origin and faces the positive direction of x-axis. Therefore, the walking speed of human doesn't affect the social positioning behaviour. Tracking the human speed is a control issue when the system tries to follow the route recommended by the social positioning and other behaviours (e.g. collision avoidance), i.e. tracking a certain desired trajectory. It is therefore not a separate behaviour but a controller to execute the decision of the behaviours.

Tracking the human or object speed has been a commonly discussed problem in robotics and many projects have studied it (e.g. Sekmen et al., 2002, Jiang and

Nijmeijer, 1997, Sun, 2005). Thus it is not our major concern in the interactive behaviour. The social positioning behaviour will, consequently, be modelled without considering human speed. We assume that the human is moving at a constant speed where by subtracting such speed from the social positioning system, he or she can be seen as stationary.

2.4 Conclusion: The Objective of the Thesis

From the review in this chapter, it can be seen that human-following is an attractive HRI scenario because it offers the challenging situation where no universal human preference exists. The robot then has to have the ability to adapt to different human preferences and to possible dynamics introduced by the interaction when it takes place in a given environment. This drives us to consider the approach of autonomous learning. The two methods that interest us are reinforcement learning and neural networks. The learning, however, has to be using-while-learning because the HRI interaction offers no knowledge about the human's preference prior to the operation of the robot system. Following the behaviour-based concept, we believe that the key to accomplishing a plausible human-following task is the interactive social positioning behaviour. This is the behaviour to position the robot appropriately according to the human's preference. It is the behaviour that needs learning and the behaviour to which this thesis will pay great attention. The primary objective of this thesis therefore is to construct a using-while-learning social positioning behaviour. The next chapter is going to further sketch the model of social positioning behaviour and to define the aim for the learning system.

Chapter 3 The Social Positioning Model

In Chapter 2, we have pointed out that social positioning is the major challenge of this thesis. The main objective is to build an autonomous using-while-learning system to accomplish the social positioning behaviour. Before any further discussion on the learning algorithm processes, the model of the social positioning scenario, on which all learning system operates, needs to be specified. Following the behaviour-based concept, as discussed in Section 2.4, social positioning behaviour is an independent behaviour that is modelled in a collision-free space where the human speed has no impact. The human can give feedback on the position of the robot, which is in the form of a performance index. The aim for the robot is to position itself based on the human feedback to achieve the best performance index possible. To establish a clear stage for the study of the learning social positioning behaviour, this chapter has two objectives. Firstly, it aims to define a minimal human preference model that provides a foundation for the learning system analysis. Then it states the target for the learning system and clarifies necessary assumptions we keep throughout the study.

In the remainder of the Chapter, Section 3.1 introduces the concept of reward surface that presents the human preference in human-following scenario and will be used throughout the simulations in this thesis. Section 3.2 introduces the aim and general concepts of the learning system in the proposed scenario.

3.1 The Reward Surface

It has been recognised by researchers that some mental function presenting the level of comfort or satisfaction in the human mind can reflect how the human thinks a task should be done. As reviewed in Section 2.1.2, modelling human behaviour and preferences plays an important role in this research. We have also pointed out in Chapter 2 that, in the scenario of human-following, the human preference differs a lot among individuals and the same individual tends to have difference preferences in different situations. Thus we adopt the concept of the existence of the unknown human mental preference function which will be referred to as the human *reward surface* in the rest of the thesis. But we aim to build the system independent of any

assumption we may make about the reward surface in the computational model and simulations so that the system is adaptive to unpredictable human preferences that are not able to be included in the pre-designed models.

However, it is necessary to make certain assumptions about the reward surface in order to compose proper simulations for the computation model and to identify some potential difficulties a human reward surface may have in the human-following scenario. Research on personal space of humans (e.g. Sack, 1986, Malmberg, 1980) points out that the preference of human can be solely based on the position of others relative to them. Therefore we can assume that the reward surface is a function of the relative position, **p**, between the robot and the human. Vector **p** is measured in the human reference frame that defines space $\mathcal{P}$. It is the frame of reference where the human is located in the origin facing the positive x-axis. The reward surface, $R \in [0,1]$, is then in the form of:

$$R = r(\mathbf{p}) \quad , \qquad (3\text{-}1)$$

where $r(.)$ is the mapping from the robot's current position to the reward surface. $r(\mathbf{p})$ is the *reward score* denoting the degree of the human's satisfaction about the robot position **p**: the higher the reward score, the higher the human's satisfaction.

In the model, the reward score is the feedback from the human, where Section 2.1.2 has reviewed some examples in literature of its detection. Although the reward score ranges from 0 to 1, the values that can be received from human while being followed vary. The levels of the reward value that they would like to use to express their feelings are different among individual cases. The more different levels a human use, the more details are described in the reward surface. We refer to those levels of reward score as *reward steps*.

At this stage of the thesis, we assume $\| \mathbf{p} \| \leq 1500$mm. This is about the width of a normal office corridor, which also limits the size of the reward surface to be 3000mm×3000mm. It is useful to mention here that the size of the map doesn't play a significant role in the learning system, the reason for which will become clear after the study in Chapter 5.

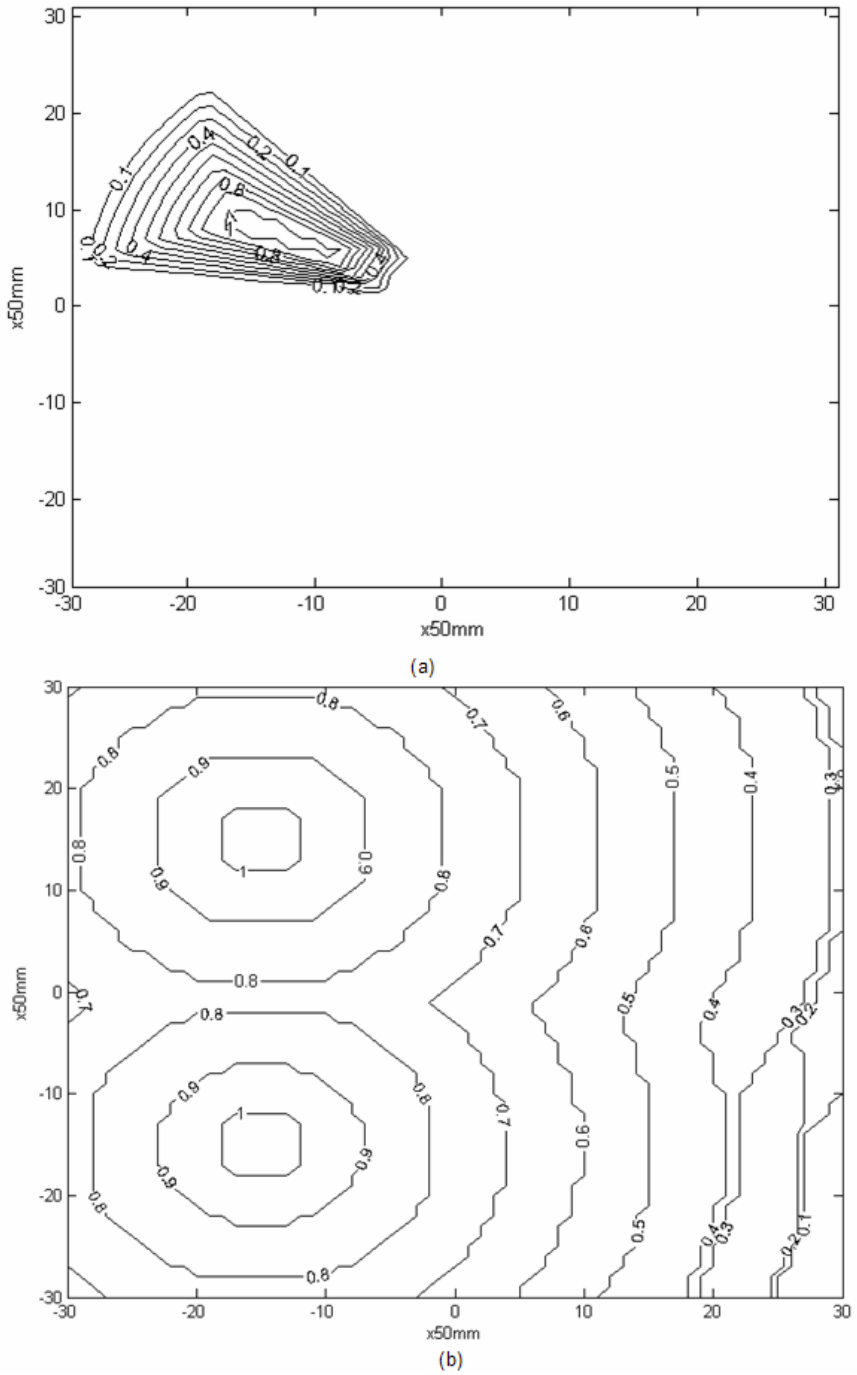


Figure 3-1 The contour plot of two possible reward surface examples: the contour lines illustrate the surface of the reward score that the human is able to provide relative to the robot's position. The values of the reward score are marked on the contour line.

Two possible examples of the reward surface are shown in Figure 3-1, both of which have 10 reward steps. In Figure 3-1(a) the target person shows a certain preference in both the angle and distance to the follower, and would therefore like to be followed behind on the left. The non-zero gradient of the reward surface only exists in a limited range, which is referred to as the *reward slope* in this thesis. The desired position is located on the reward slope, which is also referred to as the *reward*

peak in the thesis. Outside the reward slope, the surface provides a zero gradient. This part of the reward surface is regarded as *flat surface* that occupies most of the map in Figure 3-1(a). The term *flat surface* denotes the mental state of human, which provides no change in attitude. Flat surface exists because it would be unrealistic for a human to input smooth continuous feedback over the whole space. Figure 3-1(a) describes the scenario where the human is not willing to comment on all the positions over the map but those around the desired area.

Figure 3-1(b) demonstrates another situation. Two reward peaks exist in this reward surface. This happens when the corridor is narrow and the human is normally walking against the wall of one side. Then the human may have two desired places for the robot to stay as he may walk along either side of the corridor along the wall. But only half of the map, either the upper half or the lower half, is available during any one specific walk of a robot. This is a presentation of a more complex preference of human under a certain environment, which will be examined in Chapter 6.

A successful learning outcome of the behaviour interacting with human is expected to be an appropriate estimation of a movement policy desired by the human. We assume that each individual has a fixed reward surface. That is, their general preferences maintain unchanged during the operation of the robot. But even though, the human feedback can be noisy, their feelings towards the environment may change and their perception of the robot position is brief. The shape of the reward surface is by no means to remain an absolute constant. The complete feature of human preference is very unlikely to be fully included in these assumptions and unpredictable human actions in the interaction can cause changes in the reward surface during operation. Therefore the learning that has to be designed upon these assumptions can not be dependent on any specific form of reward surface. An ability to adapt to unknown and dynamic human preference is one of the basic requirements for the proposed learning system.

3.2 The Learning System

Assume that the human has an initially unknown reward surface reflecting the degree of approval of the position in which the robot is following, which will feedback into the robot system throughout the interaction. The relative position between the robot and human is known at every time step. We aim to design an autonomous online

learning robot system that learns to adapt to this reward surface and to estimate an appropriate control policy while following so that the robot will always follow the human at, or attempt to get into, the most approved position, with reasonable movements and through appropriate routes. Such a system should be able to learn and adapt to unpredictable changes in the human's preference. It is important that the system is able to operate online without pre-training because if the human preference is unknown, a pre-training is not able to be conducted.

Robot learning through action-related performance feedback from a teacher has been widely studied as an adaptive controlling approach for social robots (e.g. Bakker and Kuniyoshi, 1996, Nicolescu, 2003), where the commonly used concept is to find a close enough approximation of the desired map between the robot actions and the system states. Such an adaptive system presents the control policy as a form of mapping from selected system state space to a bounded action space. It aims to learn a general policy that reflects the desired state-action relation of the teacher and thus enables the robot to act correctly later on its own. Hirzinger (1996), describes the role of the teacher as to demonstrate successful examples where performance scores of the robot actions are provided, and a successful learning system will eventually stabilise the system on the goal state with desired actions.

The perception of spatial positions of the robot defines the system state of the proposed system. Having mentioned in Section 3.1, the state space is defined upon the human reference frame. The state space of the system is denoted by $\mathcal{P}$, and we will use $\hat{\mathcal{P}}$ to denote the quantised state space. Therefore, each possible robot position **p** belongs to $\mathcal{P}$ and is a possible system state.

The action space, $\mathcal{A}$, is the space that contains all possible robot actions. A quantised form of this space, $\hat{\mathcal{A}}$, will be defined by the transition of the states as we will do in Chapter 4. Chapter 5 will then redefine the action space based on the plant and the social requirements, e.g. safety and efficiency issues. Different definitions of the system state and action space are due to different learning methods being investigated. The learning system aims to find a close approximation of the relation, $\mu : \mathcal{P} \to \mathcal{A}$, desired by the human preference, the reward surface. The outcome of the learning system can either be a discreet map between quantised spaces, $\hat{\mu} : \hat{\mathcal{P}} \to \hat{\mathcal{A}}$, or a continuous function, $\tilde{\mu} : \mathcal{P} \to \mathcal{A}$.

One assumption that has been adopted widely in similar problems is Markov independence. A system for which the states transition is independent from previous states or actions is known to be *Markov*. This assumption has been discussed by many references (e.g. Bellman, 1957, Howard, 1960, Puterman, 1994). That is to say the desired system action is only decided by current state of the system. Markov modelling is a common assumption adopted in robot learning using reinforcement learning algorithm (e.g. Atkeson and Schaal, 1997, Pasemann, 1997) and we keep this assumption throughout our research of this behaviour.

Another condition that will be adopted throughout the learning behaviour study is that the human speed will not be considered. Chapter 2.3 has pointed out that tracking human speed is a control function outside of the robot behaviours and clarified that the social positioning behaviour takes place in the human reference frame. Thus the human speed won't have any direct impact on the social positioning behaviour and therefore the human will be considered to be still during the learning analysis. Also, leaving discussion of the robot kinematics to Chapter 7, the study and simulations before that point will assume the robot carries out the expected movement in the expected time period exactly.

We believe that there exists a mapping relating the robot position space to proper actions, which is always approved by the human and is reflected by the human feedback. Because the reward surface is not known prior to the learning, the information of the human preference, i.e. the human reward, can only be revealed while operating. More importantly, the unexplored area in the map remains uncommented by the human feedback. This requires the system to be ready for any possible new challenge when new features of the human preference are explored, although the human preference is assumed to be consistent. This then requires the system to be truly using-while-learning. Our study in the following three chapters will build an autonomous learning strategy so that the robot learns such mapping dynamically from human during the interaction and satisfies the using-while-learning requirements.

Among the literature, the learning algorithm that has been used the most to accomplish the mapping from states to actions in robotics is reinforcement learning. For systems like the proposed model that can't provide direct error measurements but only a numeric ranking score to indicate the performance of the system, the reinforcement network has the distinctive advantage of establishing the online

learning based on such a performance score with guaranteed convergence. In the next chapter, we will investigate the strengths and limits of using the reinforcement approach in the proposed scenario. It will also provide a further understanding of the proposed scenario and the learning model as well as the learning objective.

Chapter 4 Reinforcement Learning

Chapter 3 has introduced objectives of the learning system for social positioning. This behaviour aims to learn a general policy that reflects the desired state-action relation of the teacher, which can direct the robot to accomplish the assigned task autonomously. The teacher's feedback is a performance index ranking of how satisfied the teacher is with the action of the robot. Schaal(1997) identifies the learning task of such a system as: to learn such a policy that provides appropriate actions in response to a perceived state in order to steer a dynamical system to accomplish a task. The assigned task for the system is usually described in terms of optimising an arbitrary performance index without direct training data, which can't perform learning that depends on direct system error measurements such as backpropagation learning proposed by Hecht-Nielsen (1989). In Schaal's study, he proposed a reinforcement learning system to estimate the state-action map of the teacher in order to fulfil the learning task of pole balancing by a robot arm.

In the robot learning of a state-action map, the algorithm that has received the most attention is reinforcement learning, a strategy that has been adopted in many applications studying social learning of robots (e.g. Isbell et al., 2001, Calinon and Billard, 2006, Mitsunaga et al., 2005). The attraction of reinforcement learning is largely due to the absence of direct trainable system error measurements in the model. In situations where only the performance rated ranking is available, the reinforcement rule has the advantage of being able to learn, following the gradient of the reward and climbing up the reward surface (Zeidenberg, 1990). By enforcing the actions that achieve high rewards from the teacher, reinforcement learning is able to estimate a general policy reflecting the state-action relation presented by the teacher. Therefore it is worthwhile, as a preliminary endeavour, to study the reinforcement system and investigate its feasibility towards the proposed situation of social positioning behaviour.

However, a common assumption existing in published research is that every step of the robot during learning will be given trainable feedback by the teacher, because reinforcement learning generally requires a step reward measurement so that

the agent can try each action at a particular state and can evaluate its consequences immediately (Sutton, 1988) or its long-term cost (Sutton, 1990). But this requirement is hard to satisfy by a human preference. Consider the reward surface demonstrated in Chapter 3 as shown in Figure 4-1.

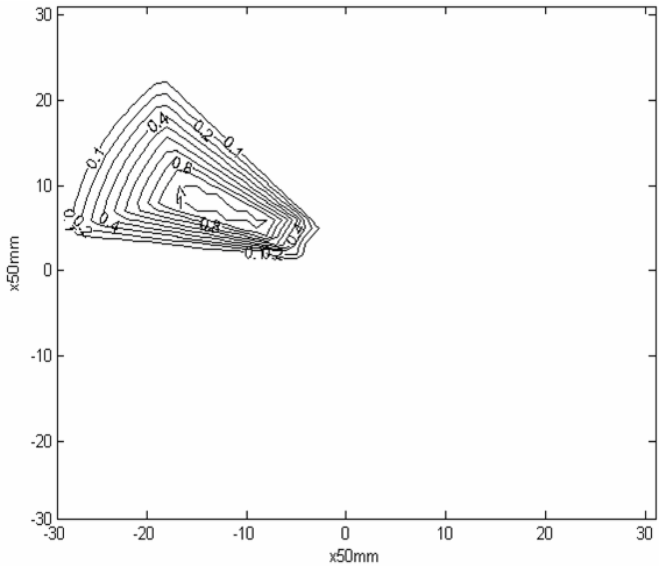


Figure 4-1 The contour plot of a reward surface representing a human's preference of being followed in the human reference frame consists of large flat surface and a small reward slope.

As shown in Figure 4-1, a large portion of the map is a flat surface. This reward surface is designed in this manner because we recognise that it is unrealistic for a human to provide a smooth and continuous feedback. In the flat surface, the human reward is not trainable information because it doesn't offer any difference among different actions and different states. In this Chapter, we introduce a secondary scoring surface which is an online-generated extension of the reward surface in the flat area to help the system perform successful reinforcement learning. In Section 4.1, we will first review one of the fundamental reinforcement algorithms: linear reward-inaction. In Section 4.2, the secondary scoring surface is introduced. The control and enhancement of learning are discussed in Section 4.3 and the simulation results are analysed in Section 4.4. Finally, Section 4.5 summarises the system's performance and discusses the further direction of the learning system that will lead our study to Chapter 5.

4.1 Reinforcement Algorithm

According to Kaelbling et al.(1996), for a system that has state space $\mathcal{X}$, action space $\mathcal{A}$ and the desired action control policy $\mu : \mathcal{X} \to \mathcal{A}$, describing the system by a discrete set of states $\hat{\mathcal{X}}$, a discrete set of actions $\hat{\mathcal{A}}$ and a reinforcement signal, i.e. the reward *R*, the reinforcement learning is to find policy $\hat{\mu} : \hat{\mathcal{X}} \to \hat{\mathcal{A}}$. This is an estimation of desired relation from states to actions, which is close enough to the target μ and satisfies *R* that is also commonly referred to as the cost or performance function.

Narendra and Thathachar (1989) point out in their survey that it is feasible to model algorithms as finite-state automata so that actions of the system describe the internal state of the system as a probability distribution according to the action that will be chosen. The probability of each action will be adjusted based on its previous failure and success. One such fundamental reinforcement rule is known as *linear reward-inaction*. It was developed based on mathematical psychology (Hilgard and Bower, 1975). Based on the reward score definition in Equation (3-1), we define ΔR^k to be the reward measurement at time *k*. For the proposed model, it is:

$$\Delta R^k = r(\hat{x}^{k+1}) - r(\hat{x}^k), \qquad \hat{x}^{k+1}, \hat{x}^k \in \hat{\mathcal{X}} \quad , \tag{4-1}$$

where superscription is the time index and $r : \hat{\mathcal{X}} \to \mathcal{M}, \ \mathcal{M} \subset \Re$ is the reward function.

Let the probability of taking action $\hat{a}$ at state $\hat{x}$ be $P(\hat{a} \mid \hat{x})$. The probabilities are assigned with small random numbers initially and the sum of the probabilities of all actions in any one state is 1:

$$\sum_{\hat{a} \in \hat{A}} P(\hat{a} \mid \hat{x}^k) = 1 : \forall \hat{x} \in \hat{\mathcal{X}} \quad . \tag{4-2}$$

For the current system state $\hat{x}^k$, the linear reward-inaction rule updates the probabilities of the actions when the measurement of reinforcement is satisfied:

If $\Delta R^k > 0$ then:

$$P(\hat{a} \mid \hat{x}^k) := \begin{cases} P(\hat{a} \mid \hat{x}^k) + \eta[1 - P(\hat{a} \mid \hat{x}^k)] & \text{for } \hat{a} = \hat{\mu}(\hat{x}^k) \\ (1-\eta) P(\hat{a} \mid \hat{x}^k) & \text{for } \hat{a} \neq \hat{\mu}(\hat{x}^k) \end{cases} : \forall \hat{a} \in \hat{\mathcal{A}}, \tag{4-3}$$

where $\hat{\mu}$ is the control policy described by the network and $\eta \in [0,1)$ is the rewarding learning rate. Learning rate is a scalar used in autonomous learning to control the strength of the step adjustment of the system while updating.

Equation (4-3) increases the probability of successful actions. In machine learning, a more appropriate measure is the expected decrease in reward gained due to executing the learning is algorithm instead of behaving optimally. Such a measure in reinforcement learning sometimes is known as *regret* (Berry and Fristedt, 1985). It penalises mistakes during the operation of the system. Patterson (1996) argues that the penalisation of linear reward-inaction rule should take place as:

If $\Delta R^k < 0$ then:

$$P(\hat{a} \mid \hat{x}^k) := \begin{cases} (1-\eta')P(\hat{a} \mid \hat{x}^k) & \text{for } \hat{a} = \hat{\mu}(\hat{x}^k) \\ \dfrac{\eta'}{n-1} + (1-\eta')P(\hat{a} \mid \hat{x}^k) & \text{for } \hat{a} \neq \hat{\mu}(\hat{x}^k) \end{cases} : \forall \hat{a} \in \hat{\mathcal{A}}, \tag{4-4}$$

where $n > 1$ is the total number of actions in set $\hat{\mathcal{A}}$ and $\eta' \in [0,1)$ is the penalising learning rate. When $\Delta R^k = 0$, the probabilities of the actions will not be updated.

Narendra (1974) proves that the linear reward-inaction rule is able to converge to an arbitrary degree if the learning rate is small enough. This reinforcement rule aims at encouraging outputs that receive a better reward score, which means, in our case, that the human is more satisfied. Though many modern algorithms have emerged after the proposal of linear reward-inaction, especially the widely used *Q*-learning proposed by Watkins (1989), we decided to apply linear reward-inaction to our model first because its simple structure allows a fast investigation but still reveals the general features of reinforcement learning. The discussion of the possibility of benefiting from a more sophisticated reinforcement rule will take place later in this chapter.

To place the reinforcement learning structure into the social positioning model, the map in Figure 4-1 has been divided into 30×30 states with fixed intervals, each of which presents a position vector, $\hat{\mathbf{p}}$. The set of discrete position vectors composes the state space $\hat{\mathcal{P}}$. At any state, the robot is able to move to any other state next to the current one or to stay at the same place. Thus there are 9 possible actions in total as the output of the network. At any position $\hat{\mathbf{p}}$, the robot can expect a

feedback reward from the human, $r(\hat{\mathbf{p}})$. By assuming that the control policy estimation $\hat{\mu}: \hat{\mathcal{P}} \to \hat{\mathcal{A}}$ is a function of current state only, the system transition $o: \hat{\mathcal{P}} \to \hat{\mathcal{P}}$ is also a function of current state because:

$$\hat{\mathbf{p}}^{k+1} = f(\hat{a}^k, \hat{\mathbf{p}}^k) = f[\mu(\hat{\mathbf{p}}^k), \hat{\mathbf{p}}^k] = o(\hat{\mathbf{p}}^k) \quad . \qquad (4\text{-}5)$$

This state transition shows that the new system state is solely decided by the current system state and independent from previous history. As discussed in Chapter 3, this is the Markov presentation of the system. The assumption that the action of the system is only dependent on the current system state remains true as long as the system is Markov.

Once an action leads to an increase in the score, $\Delta R > 0$, it is rewarded by Equation (4-3). Otherwise it is penalised by Equation (4-4). The states in the reward slope are directly trainable through Equations (4-3) and (4-4). As long as the robot is not moving in the flat surface, so that the reward score changes according to the position of the robot, it will eventually end up with a set of rewarded moves, all of which have $\Delta R > 0$.

However, if the robot is located in the flat surface, no move will cause any change of the reward, i.e. $\Delta R = 0$. This means that all actions in the flat region will not be rewarded, where no real learning can be established because no action can be rewarded by (4-3). A simple approach we propose is to use an online-generated secondary scoring surface which always has a gradient for the learning to follow.

4.2 Learning through Secondary Scoring Surface

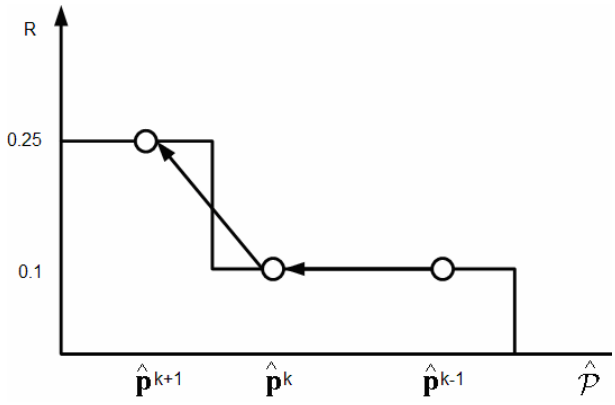


Figure 4-2 States that can't be trained by the original reinforcement rule. The horizontal axis presents a collection of states and the vertical axis is the reward value. The move starts at flat surface and manages to receive a change of reward after the action.

Figure 4-2 shows a vertical section through a reward surface with flat areas. In Figure 4-2, $\hat{\mu}(\hat{\mathbf{p}}^k)$ is an action achieving a change in the reward but $\hat{\mu}(\hat{\mathbf{p}}^{k-1})$ is not, where $\Delta R^k > 0$ and $\Delta R^{k-1} = 0$. The states that are located in the flat surface and are away from the edge of the reward slope such as $\hat{\mathbf{p}}^{k-1}$ won't learn effectively. However, it will be reasonable for them to be rewarded through (4-3) if their actions enter states like $\hat{\mathbf{p}}^k$, a state whose action will then receive a better reward score. Such learning can be established by introducing another scoring surface $r_s : \hat{\mathcal{P}} \to \mathcal{M}, \quad \mathcal{M} \subset \Re$. Initially $r_s(\hat{\mathbf{p}}) = 0 : \forall \hat{\mathbf{p}} \in \hat{\mathcal{P}}$. $r_s(\hat{\mathbf{p}})$ is updated if $\Delta R^k > 0$ and $\Delta R^{k-1} = 0$ through:

$$r_s(\hat{\mathbf{p}}^k) := (1-\eta_s) r(\hat{\mathbf{p}}^{k+1}) \quad , \tag{4-6}$$

where $\eta_s < 1$ is a very small positive constant, e.g. 0.01.

Equation (4-6) assigns state $\hat{\mathbf{p}}^k$ a secondary score that can be understood as the potential of receiving a better reward score at position $\hat{\mathbf{p}}^k$ taking the current action that has the highest probability, $\hat{\mu}(\hat{\mathbf{p}}^k)$. For all states in the flat surface, if they learn when their actions manage to receive a change in the secondary score, even if their original reward scores maintain no improvement, effective learning can still be established for the states near the reward slope. In order to expand this learning throughout the flat surface, this rule needs further extension so that every state in the flat surface is able to receive a secondary score, and the learning can be established following the change of the secondary scoring surface. At time *k*, for a position state $\hat{\mathbf{p}}^k$ in the flat surface, whose action enters state $\hat{\mathbf{p}}^{k+1}$, its secondary score is:

$$r_s(\hat{\mathbf{p}}^k) := \begin{cases} (1-\eta_s) r_s(\hat{\mathbf{p}}^k) & \text{if } \Delta R_s^k > 0 \\ r_s(\hat{\mathbf{p}}^k) & \text{if } \Delta R_s^k = 0 \end{cases} \quad , \tag{4-7}$$

where $\Delta R_s^k = r_s(\hat{\mathbf{p}}^{k+1}) - r_s(\hat{\mathbf{p}}^k)$.

After the states at the edge of the reward slope have received non-zero secondary scores based on Equation (4-6), Equation (4-7) is able to expand the slope of the secondary scoring surface over the whole flat area. Thus, after complete learning, for some state $\hat{\mathbf{p}}_a$ located at the flat surface, the secondary score will have a form of exponential descent:

$$r_s(\hat{\mathbf{p}}_a) = (1-\eta_s)^c r_s(\hat{\mathbf{p}}_b), \quad c \in \Re^+, \tag{4-8}$$

where state $\hat{\mathbf{p}}_b$ is some state adjacent to the reward edge.

For all possible states on the map, there are three possible situations:

- If the state is at the edge of the reward slope, Equation (4-6) can be used to update its secondary score if its actions enter the reward slope, where it has $\Delta R^k > 0$ and $\Delta R^{k-1} = 0$;
- If the state is located at the flat surface, Equation (4-7) can be used to update its secondary score if its actions receive a secondary score, where it has $\Delta R^k = 0$ and $\Delta R_s^k > 0$;
- To complete the rule for all states, we let the secondary score equals the original reward score while the robot is located on the reward slope, where it has $\Delta R^k > 0$ and $\Delta R^{k-1} \neq 0$.

Summarising the rules for updating the secondary score, we have:

$$r_s(\hat{\mathbf{p}}^k) := \begin{cases} (1-\eta_s) r(\hat{\mathbf{p}}^{k+1}) \text{ if } \Delta R^k > 0 \text{ and } \Delta R^{k-1} = 0 \\ (1-\eta_s) r_s(\hat{\mathbf{p}}^{k+1}) \text{ if } \Delta R^k = 0 \text{ and } \Delta R_s^k > 0 \\ r(\hat{\mathbf{p}}^k) \quad \text{if } \Delta R^k > 0 \text{ and } \Delta R^{k-1} \neq 0 \\ r_s(\hat{\mathbf{p}}^k) \quad \text{otherwise} \end{cases}. \tag{4-9}$$

The learning for the states in the flat reward surface can be established by following the change of the secondary score. This learning rewards the current action if $\Delta R_s^k > 0$ and penalises it otherwise. Therefore, for any time instance *k*, the modified learning rule can be summarised as:

- if $\Delta R^k > 0$ or ($\Delta R^k = 0$ and $\Delta R_s^k > 0$), reward current action using Equation (4-3);
- if $\Delta R^k < 0$ or ($\Delta R^k = 0$ and $\Delta R_s^k \leq 0$), penalise current action using Equation (4-4);
- update secondary score through Equation (4-9).

4.3 Learning Control

Narendra (1974) proves that the linear reward-inaction rule is able to converge to an arbitrary degree if the learning rate is small enough. However, a constant small learning rate results in slow convergence. We propose that the learning rate should change relative to the maturity of current learning action so that actions with high probability are rewarded slowly and penalised fast, and actions that have low probability are rewarded fast and penalised slowly. The calculation of the learning rate for each action is based on linear functions:

$$\begin{aligned} \eta &= H[1 - P(\hat{a} \mid \hat{\mathbf{p}})] \\ \eta' &= HP(\hat{a} \mid \hat{\mathbf{p}}) \end{aligned} \quad , \qquad (4\text{-}10)$$

where $H \in (0,1)$ is a constant that specifies the upper limit of the learning rate adjustment.

Even with the control of learning rate, the learning speed of the reinforcement network is still limited by the fact that only at most one state can learn at each step. The concept of *Neighbour learning* was introduced by Kohonen (1982) for self-organised feature maps. It is a concept often employed in competitive networks. If we assume that nearby states have similar actions, introducing such a technique can significantly increase the speed of learning by stimulating the learning of the states near the current states. For a neighbour distance *d*, if state $\hat{\mathbf{p}}$ learns action $\hat{a}$, all other states within distance *d* learn action $\hat{a}$ as well in a discounted manner. As the states of our system are actual spatial positions, it is reasonable to define the distance to be Euclidean distance.

According to Kohonen, the magnitude of neighbour learning is based on the Mexican hat function:

$$\psi(x) = \frac{2}{\sqrt{3}} \pi^{-\frac{1}{4}} (1 - \frac{d^2}{D^2}) e^{\frac{-d^2}{2D^2}} \quad , \qquad (4\text{-}11)$$

where *d* is the distance of a neighbour and *D* the limit of the neighbour learning distance. When the $D = 2$, the function appears as shown in Figure 4-3. Based on Mexican hat function shown in Figure 4-3, the learning of neighbours that have the same distance as the limit *D* has been reduced to 0 and a small region after that appears to be a penalty zone prohibiting the learning of the action fired by the current

system state. This zone is used to distinguish the recognition of different classes of patterns in the competitive network.

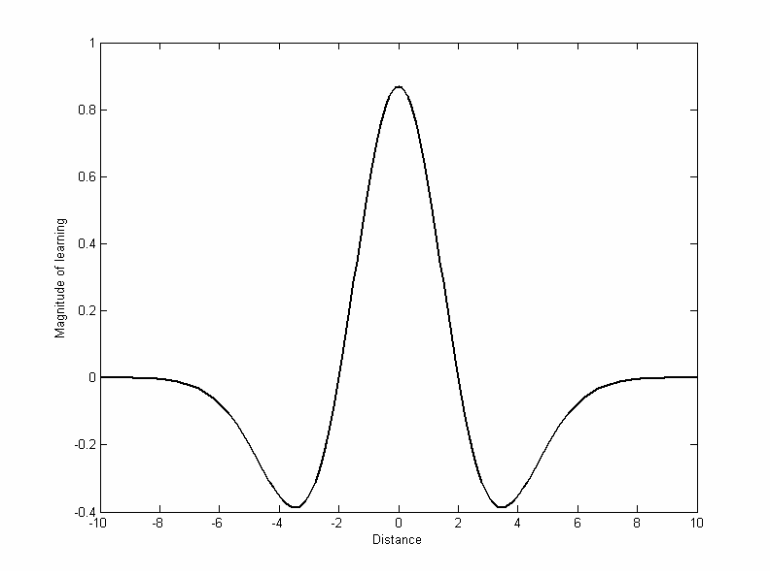


Figure 4-3 Mexican hat function with a distance of 2. When the distance large than 2, there is a prohibiting zone on each side of the x-axis, where the magnitude of learning is negative (penalising).

In the proposed reinforcement system here, there is no reason to prohibit the learning of the neighbour as the states are not competitive. Therefore, we adjusted the Mexican hat function by dropping the prohibiting factor in Equation (4-11) :

$$\psi_r(x) = \frac{2}{\sqrt{3}}\pi^{-\frac{1}{4}} e^{\frac{-2d^2}{D^2}} \quad . \tag{4-12}$$

With a distance limit of 2, the new neighbour function is shown in Figure 4-4.

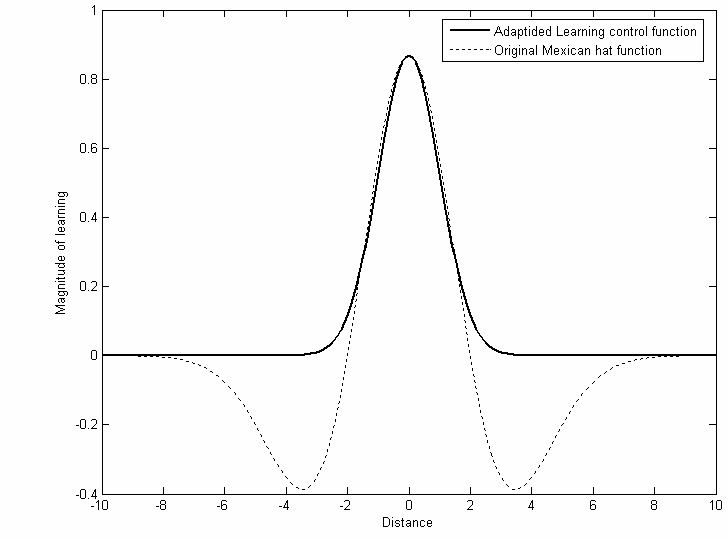


Figure 4-4 Neighbour learning function adapted from Mexican hat with on prohibiting zones and a distance of 2.

Neighbour learning allows the states of a system to interact. By assuming that nearby states fire similar actions, it can overcome the limit that only at most one state can learn in a reinforcement network. However, neighbour learning also means that a maturely learnt state can be modified by its falsely or immaturely converged neighbours, which can risk the convergence of the system. Thus, the distance limit D should be a small value. Also, neighbours learn only if they have a lower secondary score than the current learning state, which can avoid some false modification made by the neighbours.

Based on this rule, Figure 4-5 shows a comparison of a system trained by the reward surface shown in Figure 4-1 with 2000 random walks, where a *walk* denotes a set of movements carried out by the robot from a start point to the destination. The selected measurement of the system convergence is the average probability of all actions with highest probabilities: $\text{avg}\{\sum_{\hat{\mathbf{p}}\in\mathcal{P}} \max[P(\hat{a}\,|\,\hat{\mathbf{p}}) : \forall \hat{a} \in \hat{\mathcal{A}}]\}$.

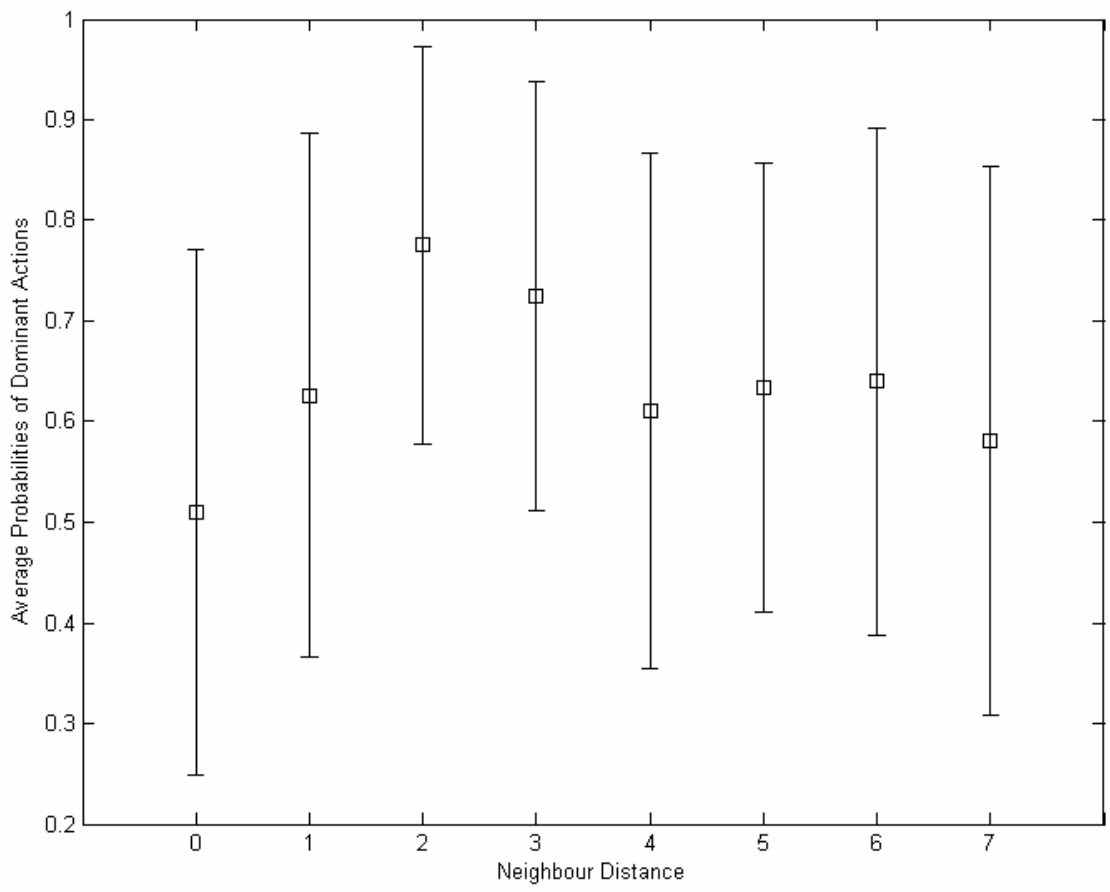


Figure 4-5 The comparison of the system convergence with different neighbour distance. Squares mark the average probabilities and the bars are the standard deviations. The value should be around 1 when the network converges ideally: the higher this value and the lower the standard deviation, the better the convergence of the system.

Figure 4-5 shows the average probability of all actions with highest probabilities after 2000 walks for networks with different neighbour distances. A well converged network ideally should have a high value close to 1 and a short standard deviation range. Test results shown in Figure 4-5 clearly support using a neighbour distance of 2 which enhances the convergence without introducing observable risk. A higher neighbour distance however has shown a risk to damage the convergence as the value of the average probability gets lower and the standard deviation gets larger.

4.4 Simulations

The simulations of the reinforcement learning system are based on the reward surface with one reward peak and a large flat surface, as shown in Figure 4-1. The aim of the system is to direct the robot to the reward peak. However, as discussed in Chapter 2, it is important for this system not only to learn successfully but also to be usable while learning. The ability of the reinforcement learning in these aspects will be investigated. The following simulations took place in the MATLAB environment.

4.4.1 Simulation System

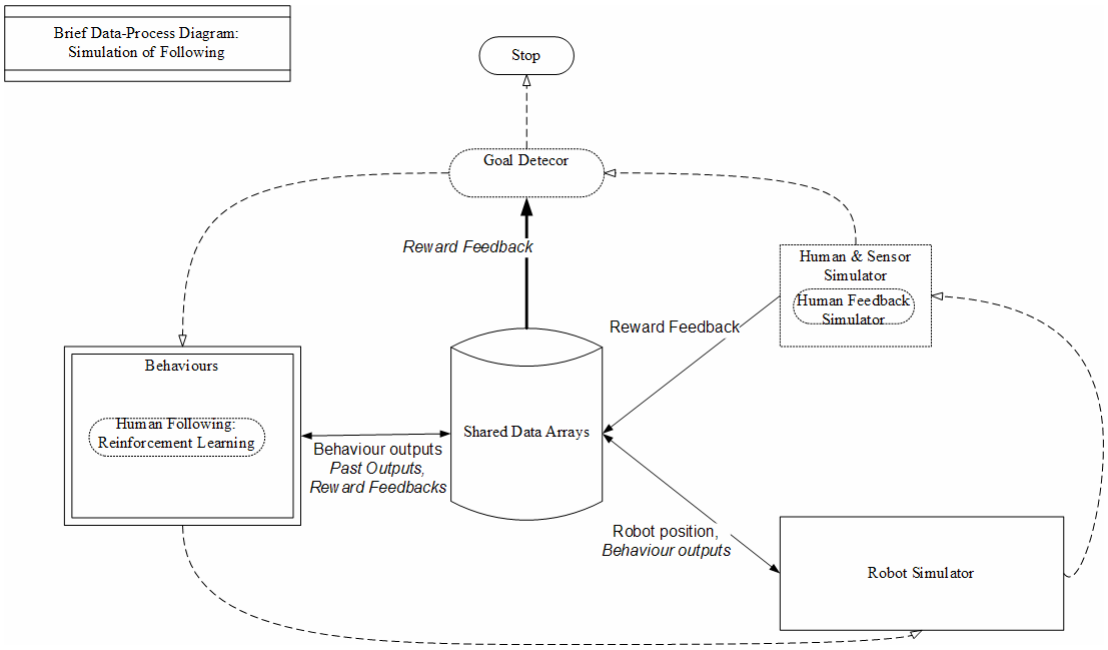


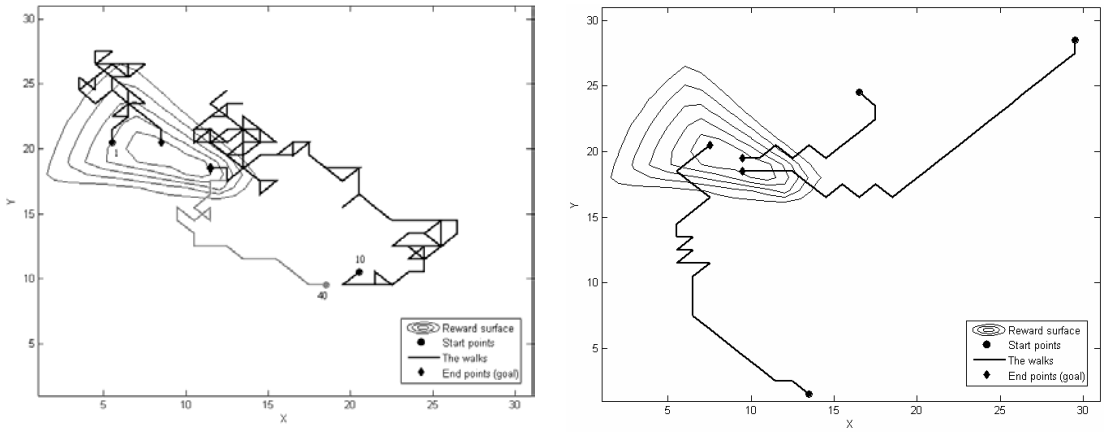
Figure 4-6 Reinforcement learning simulation system. Solid arrows represent the data flow in the system. The notations on the arrows mark the type of data in the flow, where data marked in plain form go into the shared data arrays and data with italic captions go out of it. The dotted arrows denote the flow of the execution of functions.

The simulation system is centralised around shared data. All functions follow a certain execution sequence, as shown by dotted arrows, and visit the shared data arrays in order. The goal detector stops the program when the goal is achieved. That is taken to mean that the desired position of the robot has been reached. The robot is a simulated position counter and the human feedback is the selected reward surface. As discussed in Chapter 3, the human is assumed to be stationary so their speed is not simulated in the system here.

The learning rate limit H in the system is 0.7 and neighbour distance D is 2. The constant that determines the gradient of the secondary scoring surface, η_s, is 0.05 and learning rate η is 0.1. As discussed above there are a total of 900 states with 9 possible actions for each state.

4.4.2 Simulation Results

The first simulation demonstrates the results of training based on the reward surface shown in Figure 4-1. In Figure 4-1, flat surface occupies the majority of the map. The learning therefore starts with walks that start on the reward slope so that the learning can be more effective. After 9 such walks, the training then starts at random starting points. Each walk ends after the robot reaches the goal or 1000 steps.



(a) Early walks near the reward slope. Numbers over start points mark the sequence of walks.

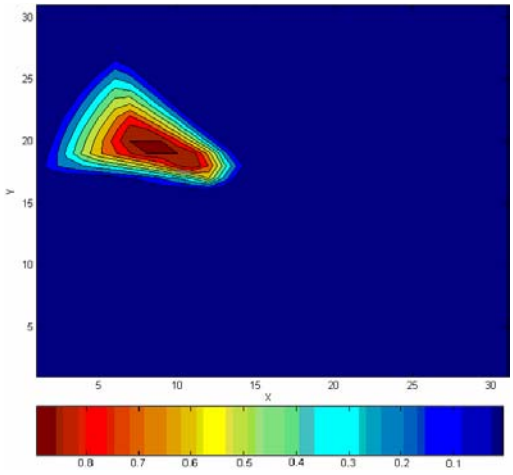
(b) Example of walks after 2000 walks

Figure 4-7 Sample movements of reinforcement system. The reward surface has one peak and large flat surface.

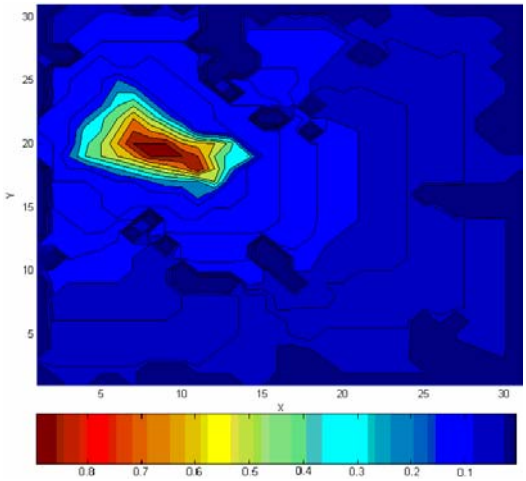
Figure 4-7(a) gives examples of the system performance during the early learning phase. The movements made by the robot appear to be aimless because most of the states in the state space have not been trained and many of the states were only

fired a few times. But given enough time, the robot can always reach the target based on the reinforcement rules. The worst possible case is that the robot achieves the goal in the early stage of learning through random wandering. The system's ability to reach the target regardless of the stage of learning is an important feature of online learning systems.

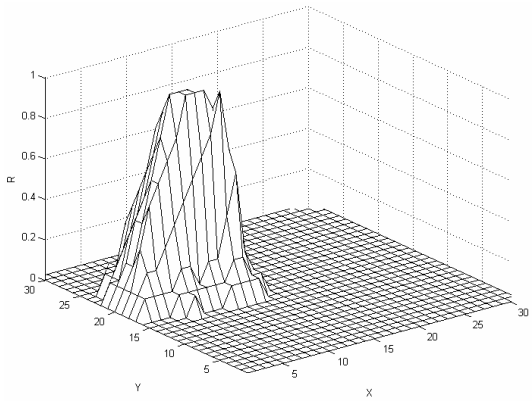
As the learning progresses, the system starts to perform more sensibly with smoother moves that move more directly to the goal, as shown by Figure 4-7(b). Those movements are then closer to the actual desired movements presented by the reward surface. Learning of the movements in the flat surface is based on the online generated secondary scoring surface. Figure 4-8 is the secondary scoring surface generated by the network after 2000 random walks.



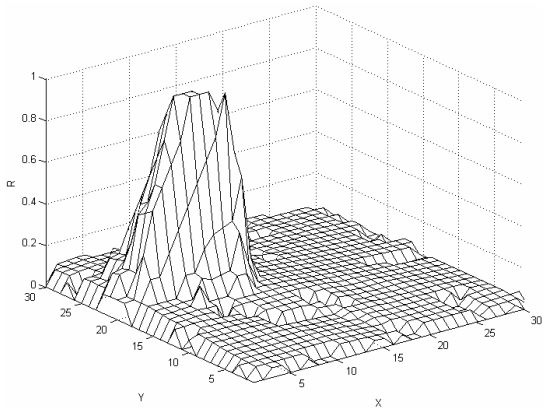
(a) Reward surface: contour plot



(b) Secondary scoring surface: contour plot



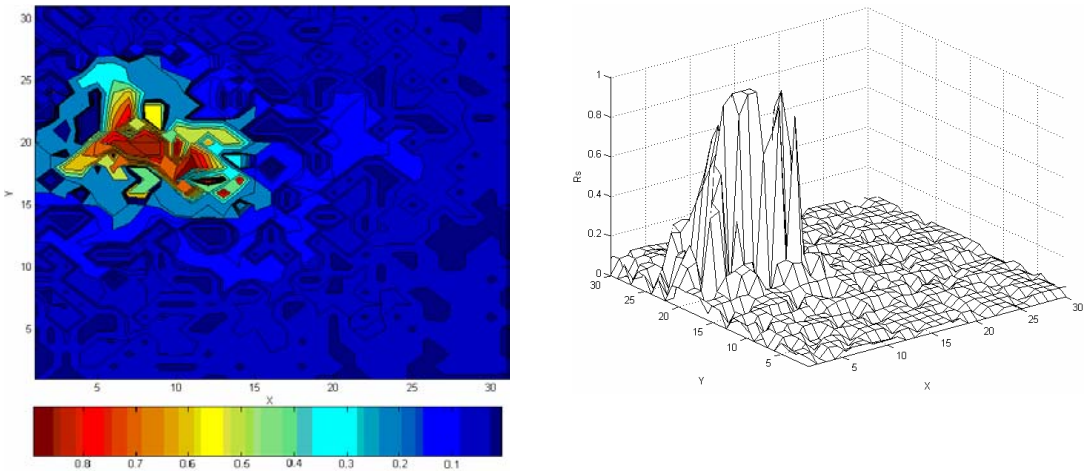
(c) Reward surface: surface view



(d) Secondary scoring surface: surface view

Figure 4-8 The comparison of reward and generated secondary reward surfaces in the case that the reward surface has a large flat surface.

As shown in Figure 4-8(b) and (d), the secondary scoring surface will not be smooth and even. The secondary scoring surface is built upon the routes taken by the robot, which contain a large number of random moves. It is an exponential descent expansion of the reward slope based on the history of the robot's movements. The gradient of the secondary scoring surface always leads to the peak of the reward surface. The deep holes that can be seen on the contour plot of the secondary surface are the locations that the robot didn't experience and the secondary scores therefore remain zero. The reinforcement system can direct the robot to the goal from any state with a non-zero secondary score in a rational manner following the generated gradient. As a further evidence of the benefits of introducing neighbour learning with small distance, Figure 4-9 shows the secondary scoring surface from the reinforcement system after 2000 walks with exactly the same setting but without using neighbour learning.



(a) The contour plot of the secondary scoring surface from a system that doesn't using neighbour learning

(b) The secondary scoring surface from a system that doesn't using neighbour learning

Figure 4-9 The generated secondary reward surface in the case that the reward surface has large flat surface and the system doesn't use neighbour learning.

The result in Figure 4-9 is based on 2000 walks of the system. It is evident that the surface is much less developed and the gradient is not direct. The gradient of the secondary scoring surface is not smooth and direct. The robot will thus use longer walks with less improved moves to reach the goal, and its paths will be worse than those presented in Figure 4-7.

4.5 Capabilities of the Network

The simulations have clearly demonstrated the network's ability to reach the most desired position in a finite number of movements. The system works online in that the goal can be reached without providing any pre-training. The system starts by making random movements and the performance of the system gradually becomes more direct and smoother. The gradient of the secondary scoring surface is an exponential descent of the reward slope and the states with non-zero secondary score converge to the actions that can follow the generated gradient to the goal. As the learning progresses, the secondary scoring surface spreads the slope throughout the state space. It is generated online based on the data collected over time.

However, the nature of reinforcement learning is probability counting, which has to be based on a significant number of random actions. It works online because reinforcement learning's convergence rule is based on statistical theory and, given enough time, the system can always stabilise at the goal, regardless the stage of learning. However, the early stage of the training has to be filled by a large number of random moves, which is hardly feasible for any real robot motor system. This chaotic performance is unavoidable because unlearnt states can only fire randomly. The system improves the performance as learning progresses but the process is lengthy. After a robot has followed a human two thousand times, the human would expect a behaviour better defined than the results presented in the simulation. Infeasible early output, lengthy learning processes and a less-than-ideal learning outcome make it hard to use this learning online to control the robot in real world. The simulation results show that the reinforcement learning system here is not using-while-learning.

There are reinforcement rules known to produce better learning outcomes. One of the most popular algorithms is *Q*-learning proposed by Watkins(1989). It has been extensively used in a wide range of areas. A large number of examples of robot learning using *Q*-learning can be found in the literature (e.g. Schaal, 1999, Nicolescu, 2003). But, as a form of reinforcement network, it still relies on excessive random actions to start the learning. By describing the system as a collection of discrete states, the reinforcement system is not able to function acceptably until the majority of the states in the collection have converged to a correct action. 'By trying all actions in all states repeatedly, it learns which are best overall' (Watkins and Dayan, 1992). The online use of *Q*-learning or other form of reinforcement rules on real robot motor

systems thus has been strictly limited to those systems with a small state space and a limited number of initial states (e.g. Rummery and Niranjan, 1994, Pasemann, 1997) so that the system can experience all possible states within a small amount of time. But the most common practice with *Q*-learning in many related robot applications is to use long pre-training sessions until a plausible state of the system has been achieved (e.g. Nicolescu et al., 2006, Calinon and Billard, 2006). By using rules such as *Q*-learning, one can be confident that the final outcome of the system will be closer to the ideal function but there is little evidence to support its improvement in the performance of the early learning stage and the time of learning compared to the proposed linear reward-inaction. For reinforcement learning, unexpected states remain unlearnt and when the system steps into such a state, the system relies on random moves to start the learning. To experience almost every possible state becomes the prerequisite for a reinforcement network to provide acceptable performance. However, it is unrealistic for the proposed human-following scenario.

Reinforcement learning lacks the ability to produce appropriate outputs for novel states based on previous learning. Such an ability is known as *generalisation*. It is essentially the process of approximating a function by estimating the relation between the input and output space given limited examples from the true target function. Networks that can generalise only need to experience a limited number of states to conclude a possible outcome for unlearnt states. The system then is no longer described as sets of finite state and action collections but as a continuous function relating the position of the robot to the actions. A widely used learning algorithm for neural networks, error backpropagation, is known to be able to generalise with the given sample points acting as teacher, i.e. training data. However, recalling the summary of the problem modelled here from Schaal (1997) at the beginning of this chapter, such training data don't exist in the model used in this thesis.

Researchers have been studying possible improvements of generalisation of reinforcement learning (e.g. Sutton, 1996, Boyan and Moore, 1995, Ackley and Littman, 1990). However, because the basic reinforcement learning rule doesn't generalise, researchers have to introduce the generalisation into the system by compromising some factors of the system, typically the learning speed. That is to say, by introducing generalisation into reinforcement learning, it produces an even lengthy training process as the reinforcement needs longer learning time than usual to produce a generalisation. This is acceptable for those research projects because most of their

projects are carried out in offline learning scenarios. However, the learning speed can not be compromised in our scenario.

Generalisation is an important and attractive feature for many robot learning tasks. In social learning, generalisation provides the ability to conduct behaviours in novel situations based on previous experience. Therefore, in the next chapter we will explore and investigate the alternative of building an online learning network that generalises and overcomes the absence of training data, based on a well studied neural network structure: the multilayered feedforward network.

Chapter 5 Multilayered Feedforward Networks using Online Backpropagation

In Chapter 4 we have studied one of the possible learning algorithms for the autonomous learning behaviour, reinforcement learning. It aims to find the most appropriate location to follow the human through reasonable paths, according to human feedback. The reinforcement algorithm chooses a set of discrete states, with a set of discrete actions. Through learning, the reinforcement system establishes a discrete map between the state and action sets. By rewarding and penalising actions based on the feedback to the system, an approximation of the true mapping between the states and the actions can be expected. For a close approximation, however, a large number of states and actions are needed. Large state and action sets generate unfeasible initial outputs because the system has to rely on random firing neurons for a long time and the time to converge will be long. Therefore, in order to produce an acceptable performance, reinforcement learning needs to experience all states of the system repeatedly. Reinforcement learning lacks the ability to generalise. Generalisation is a frequently mentioned term in autonomous learning, which refers to the system's ability to estimate the output of novel untrained states based on given examples. With non-experienced states remaining unlearnt in a reinforcement network, the long period of random firing and slow convergence is unavoidable. However, for a reliable online learning system to control the movement of the robot, it is necessary for the learning system to provide a plausible control policy fast enough in its initial training for a real using-while-learning operation.

Instead of finding a discrete mapping between the quantised input and output spaces, an alternative is to approximate the true mapping between the state space, $\mathcal{X}$, and action space, $\mathcal{A}$, by assuming that differentiable function exists, which is close enough to the true relation between the two spaces: $\mu : \mathcal{X} \to \mathcal{A}$. For a neural network that defines a system function, $\tilde{\mu} : \mathcal{X} \to \mathcal{A}$, it is possible to build it up as an approximation to μ to a satisfactory degree online, given some sample data from the true relation, μ, during the operation. One such learning algorithm for a neural

network working in this manner is backpropagation proposed by Hecht-Nielsen (1989), and now sometimes is more specifically referred to as *first order error gradient descent backpropagation* (Battiti, 1992).

Backpropagation learning has been widely adopted across a wide range of engineering applications. It is attractive because of its ability to act as a universal function approximator (Hornik et al., 1988). Although most of the applications in the literature are based on offline function approximation, its online learning ability has been introduced into many research projects, typically in control system applications (e.g. Zhao et al., 2002, Chen, 1990, Ahmed et al., 1995). However, few examples were found in literature applying backpropagation to a robot behaviour that learns by interaction with the human. The challenge of using this learning algorithm in the proposed HRI model is finding the error measurements of the system outputs. Because the human preference is unknown to the system, the desired system outputs that have to be used to measure the system error can't be collected. The learning of this scenario, therefore, has been investigated mainly using reinforcement learning, as reviewed in the previous chapter. This chapter will propose a novel online learning method to enable backpropagation learning for the robot behaviour in the HRI scenario, so that the system can benefit from the advantage of generalisation.

In the remainder of this chapter, Section 5.1 reviews the backpropagation algorithm and the structure of multilayered feedforward neural networks. Section 5.2 proposes the online training data selection method to form the system error measurement during operation. Section 5.3 introduces the proposed online backpropagation algorithm and neural network structures. Section 5.4 demonstrates the simulation results of the system and provides a deep analysis of the system performance. Section 5.5 is a short conclusion of the study in the chapter and opens the issues for further investigation in Chapter 6.

5.1 Introduction of Error Gradient Descent Backpropagation and Multilayered Feedforward Neural Networks

Error gradient descent backpropagation is one of the fundamental learning rules in neural networks. Its preliminary form was discovered by Bryson and Ho (1969), and Werbos (1988). Its modern formula was proposed by Hecht-Nielsen (1989), who also proved the convergence of the learning algorithm. The backpropagation algorithm

was originally proposed for a classic branch of neural networks, now known as *multilayered feedforward neural networks*, a useful overview of which has been given by Bebis (1994). Backpropagation has also been used in other forms of neural networks that were developed later such as *recurrent neural networks* for closed-loop systems (e.g. Stroeve, 1998, Sutton, 1990, Steil, 2005). Recurrent networks are mainly used for time-related mapping, where the current state of the system is expected to be related to some or all previous states. In other words, the system is non-Markov. Keeping the Markov assumption (see Section 3.2), the multilayered feedforward neural network is an appropriate choice.

In the work of Hecht-Nielsen, he not only refines the backpropagation algorithm but also proposes the modern connectionism of the multilayered feedforward network. It is a neural network with a hierarchical design consisting of fully interconnected layers of neurons. The structure of the multilayered feedforward neural network is shown in Figure 5-1.

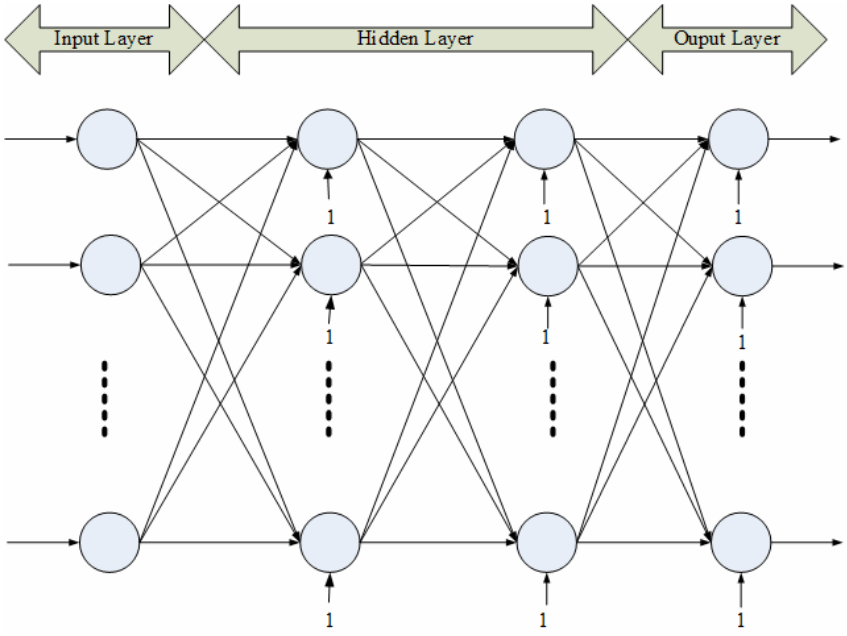


Figure 5-1 The architecture of the multilayered feedforward neural network. It consists of one layer of input units, one layer of output units and several layers of hidden units. The adjustable weights are associated with all connections between neurons. Each non-input neuron has an adjustable threshold, or bias, represented by an adjustable connection with unit input.

In general, a multilayered feedforward neural network contains one input layer, one output layer and several layers in between, called *hidden layers*. All neurons are fully connected to the neurons in the previous and next layers, if there are any. Each non-input neuron has a connection with unit input, called *bias* or *threshold*. Given a set of examples known as *training data* that specify some sample inputs and

their desired outputs, the backpropagation rule adjusts the weights of the network autonomously. Through this adjustment, the system function is able to converge to an approximation of the target function to an arbitrary degree of accuracy.

Based on Hecht-Nielsen's proposal of backpropagation, in a multilayered feedforward network, $w_{ij}(k)$ is the weight of the connection between i^{th} neuron in j^{th} layer and k^{th} neuron in $(j-1)^{\text{th}}$ layer. Define $\mathbf{w}_{ij} \equiv [w_{ij}(1), w_{ij}(2), \cdots, w_{ij}(n_{j\text{-}1})]^{\text{T}}$ that denotes the weight vector of the i^{th} neuron in j^{th} layer, and $\mathbf{x}_{ij} \equiv [x_{ij}(1), x_{ij}(2), \cdots, x_{ij}(n_{j\text{-}1})]^{\text{T}}$ to be the input vector of the neuron, where $n_{j\text{-}1}$ is the number of the neurons in layer j-1. Let $y_{ij} = \mathbf{w}_{ij}^{\text{T}}\mathbf{x}_{ij} + b_j$, known as the net value of the neuron, where b_j is the weight of the bias connected to the i^{th} neuron of j^{th} layer. The output of the neuron is a transition of the net value:

$$z_{ij} = f(y_{ij}) \quad , \tag{5-1}$$

where $f(.)$ is known as the *activation function*. Early research on backpropagation and neural learning (e.g. Hopfiled, 1984, Grossberg, 1982, Williams, 1983) advocated the activation function to be a sigmoid function, and it has been adopted ever since. The choice of activation function has now widened to a series of sigmoid-type functions (Huang and Babri, 2000, Menon et al., 1996).

For the input layer, the activation function is linear and it has no input connections with adjustable weights. It actually serves simply as the memory for input vectors and sometimes researchers prefer not to call it a layer of the system. For the sake of clarity, when the number of layers of multilayered feedforward networks is mentioned in this thesis, it always assumes that the input layer is numbered as layer one.

According to Hecht-Nielsen, the aim of updating of the weights is to adjust the function presented by the network autonomously so that a certain performance function is minimised. The performance function is a measurement of the difference between the system outputs and the desired outputs for the given examples. The training data set is a collection of examples of some sample inputs and corresponding desired outputs of the target function, μ. The error function, **E**, is calculated so that the system works in a least square manner:

$$\mathbf{E} = \frac{1}{2}\sum[(\mathbf{t}-\mathbf{z})\cdot(\mathbf{t}-\mathbf{z})] \quad , \tag{5-2}$$

where **z** is the system output vector and **t** is the corresponding vector of desired targets.

The system weights are assigned small random numbers initially. In backpropagation, the adjustment of the weights should be proportional to the descent of the system error gradient. Given that the weight matrix of the j_{th} layer $\mathbf{W}_j \equiv [\mathbf{W}_{1j}, \mathbf{W}_{2j}, \cdots, \mathbf{W}_{n_j j}]$, it should be updated as:

$$\Delta \mathbf{W}_j = -\eta \frac{\partial \mathbf{E}_j}{\partial w} = \eta \boldsymbol{\delta}_j \mathbf{y}_j^T , \tag{5-3}$$

where $\eta \geq 0$ is the learning rate, $\mathbf{E}_j$ is the performance function of the j^{th} layer, and $\mathbf{y}_j \equiv [y_1, y_2, \cdots, y_{n_j}]^{\text{T}}$. $\mathbf{E}_j$ is unknown except for the output layer as the expected output for hidden neurons can't be decided. Thus $\boldsymbol{\delta}$ is introduced to backpropagate as the system error through the hidden layers, the value of which is:

$$\boldsymbol{\delta}_j = \begin{cases} (\mathbf{t} - \mathbf{z}) \cdot \dot{\mathbf{z}}_j & \text{for} \quad j = h \quad ; \\ (\mathbf{W}_j^T \boldsymbol{\delta}_{j+1}) \cdot \dot{\mathbf{z}}_j & \text{for} \quad 1 < j < h, \end{cases} \tag{5-4}$$

where h is the total number of layers and $\dot{\mathbf{z}}$ is the vector that consists of the derivatives of each element in vector **z**.

The autonomous learning of the network is achieved by updating the weights based on the selected performance function, **E**. However, the proposed HRI model lacks the measurement of system error performance. The training data set doesn't exist because human preference, which is presented by the reward surface, is unknown to the learning system and the desired outputs of the system, i.e. the preferred robot movements by the human, can't be pre-collected. Thus collecting training data is a prior condition for the construction of a multilayered feedforward neural network. A novel process of collecting useful data to form a set of training data online is therefore proposed.

5.2 Training Data Selection

Human feedback is used to rank the performance of the robot's movements while operating. Based on human's responses, it is possible to collect a set of moves that have achieved positive feedback from the user. Positive feedback doesn't mean the robot has made the best possible move. However, if the system is trained by these data, its performance can be expected to improve. A better performing system is then

able to collect data that are closer to the ideal human preference. The data then further refine the system performance. In such a way, it is possible to base this behaviour on backpropagation learning online without prior training data.

For the social positioning behaviour here, the system state space, $\mathcal{P}$, is the space that contains all possible positions of the robot relative to the human. The input for the system in the proposed model is the instantaneous robot position $\mathbf{p} \in \mathcal{P}$. The output of the system, $\tilde{\mu}(\mathbf{p})$, is the expected action of the robot at the next time instance, which is assumed to be carried out exactly at this stage (see Section 3.2). The action is in the form of an expected displacement. The system transition is therefore:

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \tilde{\mu}(\mathbf{p}^k) \quad . \tag{5-5}$$

The reward measurement, $\Delta R^k = r(\mathbf{p}^{k+1}) - r(\mathbf{p}^k)$, is essentially the same as Equation (4-1). $\Delta R^k > 0$ means that the current move $\tilde{\mu}(\mathbf{p}^k)$ receives the human's approval. Though it may not be the best move, if $\mathbf{p}^{k+1}$ is the best rewarded place that the robot has achieved so far by making a move from position $\mathbf{p}^k$, it is sensible to train the system with the input pattern $\mathbf{p}^k$ and desired output $\tilde{\mu}(\mathbf{p}^k)$. Because the learning of the system improves the moves, the chance of better movements happening at $\mathbf{p}^k$ exists, which can replace the old training data. In this way, the data selecton and system learning enhance each other through time.

Thus the system output that has received the greatest increase in reward up to the current time at each position is taken as the target for training with the current input pattern. To store the collected training data in a finite amount of memory, a discrete state set $\hat{\mathcal{P}} \subset \mathcal{P}$ is used which presents the position space in 60×60 states with fixed intervals between them. We define $\Delta r_t(\hat{\mathbf{p}}) : \hat{\mathcal{P}} \to [0,1]$ to be the best reward measurement collected by making a move at state $\hat{\mathbf{p}}$, and $\mu_t(\hat{\mathbf{p}}) : \hat{\mathcal{P}} \to \mathcal{A}$ to be the action that acquired that reward measurement. Initially, $\Delta r_t(\hat{\mathbf{p}}) = 0,\ \mu_t(\hat{\mathbf{p}}) = [0,0]^{\mathrm{T}} : \forall \hat{\mathbf{p}} \in \mathcal{P}$. After data are collected, a trainable input-target data pattern is formed as $\{\hat{\mathbf{p}},\ \mu_t(\hat{\mathbf{p}})\}$. Δr_t is used to validate if new data patterns are better and should replace the existing ones. The primary updating rule of the training data collection is:

if $\Delta R^k > \Delta r_t(\hat{\mathbf{p}}^k)$:

$$\begin{aligned} \mu_t(\hat{\mathbf{p}}^k) &= \tilde{\mu}(\mathbf{p}^k) \\ \Delta r_t(\hat{\mathbf{p}}^k) &= \Delta R^k \end{aligned} \quad , \tag{5-6}$$

where $\hat{\mathbf{p}}^k \in \hat{\mathcal{P}}$ is the closest state to position $\mathbf{p}^k$.

If $\Delta R^k < 0$, we have $r(\mathbf{p}^{k+1}) - r(\mathbf{p}^k) < 0$ or, equivalently, $r(\mathbf{p}^k) - r(\mathbf{p}^{k+1}) > 0$. It suggests that going from $\mathbf{p}^k$ to $\mathbf{p}^{k+1}$ is not an appropriate move. However, if the robot were moving backwards, from $\mathbf{p}^{k+1}$ to $\mathbf{p}^k$, it would experience an increase in the reward. If $\mathbf{p}^k$ is the best rewarded place that the robot has achieved by taking a move from $\mathbf{p}^{k+1}$, $\mathbf{p}^k$ is a trainable pattern with a desired movement, $-1\times\ \tilde{\mu}(\mathbf{p}^k)$. Thus:

if $-1\times\Delta R^k > \Delta r_t(\hat{\mathbf{p}}^{k+1})$:

$$\begin{aligned} \mu_t(\hat{\mathbf{p}}^{k+1}) &= -1\times\tilde{\mu}(\mathbf{p}^k) \\ \Delta r_t(\hat{\mathbf{p}}^{k+1}) &= -1\times\Delta R^k \end{aligned} \quad . \tag{5-7}$$

Equation (5-7) is considered as the secondary means of data selection because, as the learning progresses, moves that cause ΔR^k to be negative happen less frequently and the collection of training data is mostly the duty of Equation (5-6). The rule stated by Equation (5-6) and (5-7) updates the training data when a change of the reward score happens, i.e. $\Delta R^k \neq 0$. As a result, the states located in the flat surface won't have any training data selected. But, because the backpropagation is able to generalise reasonable outputs for untrained states based on the learnt examples, an overall control policy for the robot can be expected after effective training.

5.3 Online Backpropagation Learning

5.3.1 Neural Network Structure

It is evident that the size of the network is a significant factor that affects the rate of convergence (Gao and Yang, 2003). The optimal number of hidden layers and neurons has been a common topic of discussion in the literature. Unfortunately, no formula or hard criteria have been proposed to determine the optimal size of the network. Existing approaches propose repeated experiments with statistical comparison to find out the best size for offline learning (Kim and Yum, 2004), typically *signal noise ratio analysis* (e.g. Khaw et al., 1995, Yang and Lee, 1999). But these methods are targeting the best convergence of offline learning, the results of

which are difficult to apply in the online learning here without a constant set of training data.

With no practical methods to decide the best network scale beforehand, the most common practice among existing online learning in engineering applications is to adopt a middle-sized network for a learning task whose target is known not to be too complex. The size of those networks is normally no more than 10 hidden neurons in each hidden layer and with 1 or 2 hidden layers (e.g. Zhao et al., 2002, Chen, 1990, Ahmed et al., 1995). Considering the importance of the speed of learning in a using-while-learning operation, we chose a network with one hidden layer with 10 neurons.

A successful system requires, however, that the number of hidden neurons is enough to map the target function. In order to have a small network, it is useful to simplify the learning target for each network so that the system can learn fast and successfully. The method proposed is to separate the system into two networks, one for the movement in the x direction and one for the y direction in the human coordinate frame. The movements in x and y are orthogonal therefore separating them is mathematically plausible. Because the hidden neurons in each network are then only responsible for satisfying one output rather than compromising between two outputs, smaller sized networks are possible, as the two outputs no longer need to share the hidden neurons. The simulation in Section 5.4 will prove that this is an effective and sufficient configuration.

5.3.2 Backpropagation Learning

Using the data selection method, the backpropagation learning can be performed. The selected system performance function is the sum of squared error:

$$\mathbf{E}_a = \frac{1}{2} \sum_{\forall \hat{\mathbf{p}} \in \mathcal{P}, \Delta r_t(\hat{\mathbf{p}}) > 0} [\frac{\mu_t(\hat{\mathbf{p}}) - \tilde{\mu}(\mathbf{p})}{K}] \cdot [\frac{\mu_t(\hat{\mathbf{p}}) - \tilde{\mu}(\mathbf{p})}{K}], \quad (5\text{-}8)$$

where $\hat{\mathbf{p}} \in \hat{\mathcal{P}}$ is the closest state to the input position $\mathbf{p}$ in system memory, and scalar K is the limit of the robot speed. The speed limit K is taken to be 500 mm/s. It is a safe robot speed in a social environment and a robust range for the motor controller (details in Chapter 7.4). The error function is normalised because the output of the neural network belongs to the range [-1, 1]. The activation function for the hidden layer in our system is:

$$f(x) = \tanh(\frac{x}{\alpha}) = \frac{e^{x/\alpha} - e^{-x/\alpha}}{e^{x/\alpha} + e^{-x/\alpha}} \quad , \tag{5-9}$$

where $\alpha \in \Re^+$ is a constant. α decides the width of the slope in the tanh function. Researchers, such as Dundar (1998), have reported that different settings of activation function affect the learning of the network. The tanh function is used because the output of our system is bipolar. In the research of Gao and Yang (2003), they argue that 0.3 is likely to be an effective value for α giving a reasonable slope under a similar setting of the network. This value is adopted in our network.

A common practice for offline backpropagation learning rules is to use a learning rate that decreases through time (Patterson, 1996), in order to improve the convergence. It is inappropriate in the online learning here as the training data set keeps updating and any trainable data can be new. Thus the significance of learning is not marked by the time indices but by the novelty of the training pattern, which is associated with the error performance of the system: higher novelty leads to worse error performance. Therefore, the learning rate is calculated online based on the mean of system performance:

$$\eta_a = H(1 - e^{-\beta\sqrt{\frac{E_a}{m}}}) \quad , \tag{5-10}$$

where m is the number of error measurements, e is the natural logarithm constant, $H \in \Re^+$ is the learning rate limit and β is a constant specifying the speed of the descent of the learning rate relative to the decrease in system errors. β is 0.8 in the simulations, which gives a reasonable change of learning rate when $\frac{E_a}{m}$ falls in [0, 4]. $\frac{E_a}{m}$ is bounded in [0, 4] because the network output is limited in [-1, 1], decided by the tanh function.

5.4 Simulations and Analysis

The simulation system is the same as the one used in Chapter 4 except that the reinforcement learning behaviour is replaced by the backpropagation learning behaviour. This section demonstrates general results of the system behaviour and provides an in-depth analysis of its operation, under the scenario of locating the appropriate position in the human coordinate frame based on the reward feedback.

The training was carried out in a pattern-by-pattern basis. The data used in this section will be selective and representative. The complete system performance history of all the system operations mentioned in this section is located in Appendix I.

5.4.1 Simulation Results

The first simulation trains the system with a reward surface that contains one peak and a large flat surface. Because the training data are collected online, it is important for the system to have enough useful feedback from early operations. Therefore, the first 3 walks start on the reward slope and the robot starts at random points afterwards. For each walk, the system operates for 400 steps regardless of its failure or success at the end of the walk, which provides a fixed time window for the analysis. The system parameters and structure are as discussed in Section 5.3. Some walks of the system are shown in Figure 5-2. The weights of the system are randomly initialised before the start and they will not be reinitialised in the same simulation. Incremental backpropagation training was practiced.

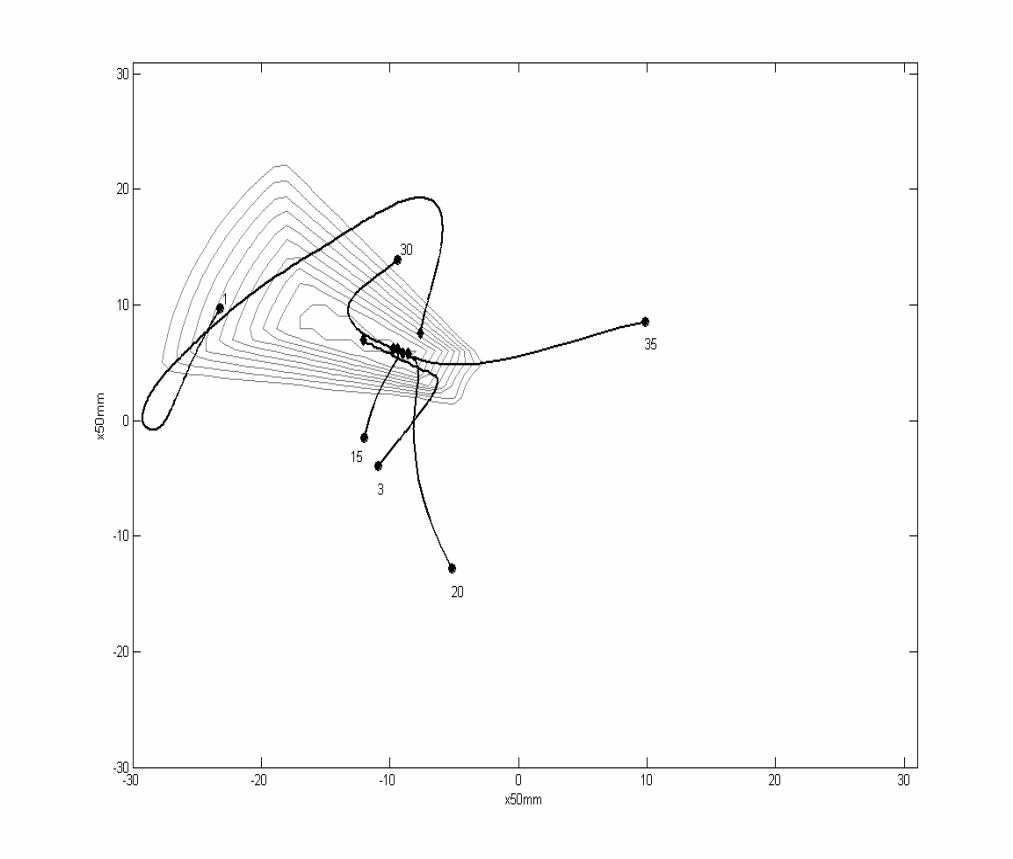


Figure 5-2 The walks produced by the online learning multilayered feedforward networks with backpropagation. The target reward surface has one reward peak and a large flat surface. The ● marks the starting point of the robot and ♦ marks the ending/stabilising point of the robot. The numbers indicate the index of the walk in the operation.

The demonstrated walks have shown that later walks have a clear improvement over early ones in terms of the paths that they generate. It is promising that the system is able to direct the very early walks to the goal. The walks that start on the flat surface also locate the goal correctly. This demonstrates the system's ability to generalise. Such generalisation is based on the reward surface, in that the walks are trying to follow the steepest gradient of that surface. The 30th walk in Figure 5-2 is a good example of the effect. Instead of going directly to the reward peak, the path given by the 30th walk is obviously following the gradient of the reward slope.

It is shown in Figure 5-2 that the first walk ends before the goal is reached. This is because the 400-step time limit has been reached. Given long enough time for the system, under the scenario set up for this simulation, failure is rare and might only happen if the robot starts at the edge of the reward slope and heading in the wrong direction with a very small velocity at the very first walk. In this situation, the robot can only collect one set of training data with a small value before it enters the flat surface. Because both the values of the system output and the training data are small, the error measurement is small and so are the weight adjustments. The robot will then go out of the map before it can turn back.

Normally after 40-50 walks, further learning causes little observable improvement in the system performance. Although the system learns based on the reward surface, its direct learning outcome is actually the expected displacement at the next time index for each position in the map. The system function, $\tilde{\mu}^{2\times 2} : \mathcal{P} \to \mathcal{A}$, is a continuous 4-dimensional surface. Surveying a set of positions in the map and their outputs, the system function can be illustrated as Figure 5-3, as a quantised discrete illustration of the 4-dimensional mapping.

Figure 5-3 demonstrates the generalisation of the system clearly. No training data can be collected in the flat surface, but a reasonable set of movements have been produced. It can be seen that a reasonable movement surface has a good similarity to the gradient of the reward surface in the regions in which the non-zero reward gradient exists. The integration of the system output, therefore, can indicate the system's recognition of the reward surface, and it will be worthwhile to compare it with the reward surface to evaluate of the system's behaviour.

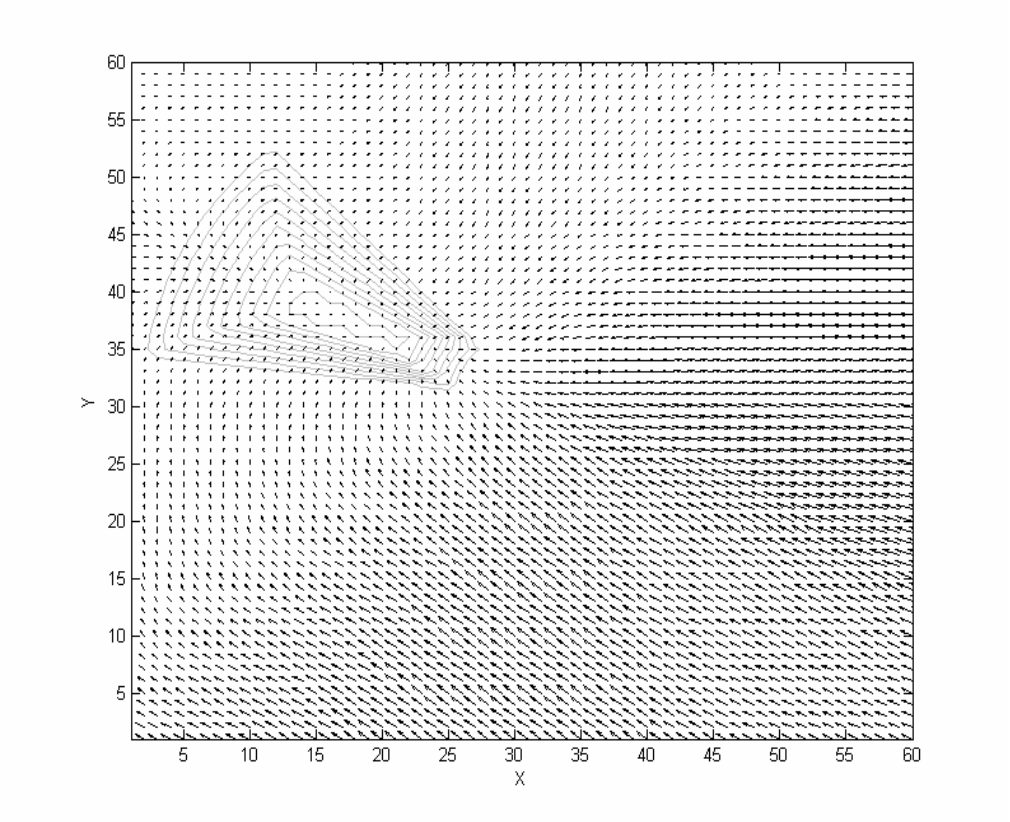
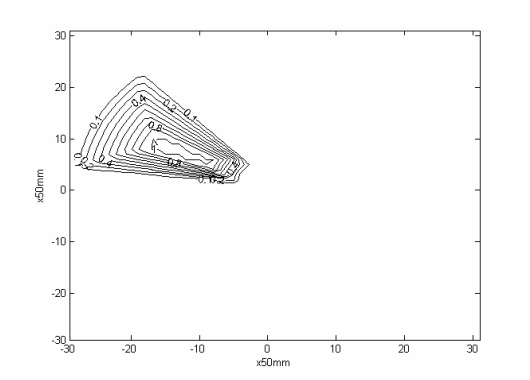


Figure 5-3 The multilayered feedforward system performance after 30 walks of online operations: the arrows indicate the system output speed vector of the movements at sample positions.

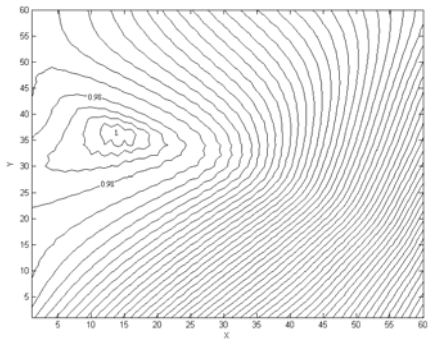
Integrating the movement surface with finite difference integration[1], the reward surface implied by the system output after 30 walks is shown in Figure 5-4(b) with a comparison with the reward surface in Figure 5-4(a).

The integrated output surface has been normalised into the range [0, 1]. It has to be noticed that for data that don't contain constant gradient, such as the system outputs here, the integration algorithm here can only work in a least square sense and a certain bias may exist in the outcome. Therefore, the integrated output surface can only serve as a point of reference but not an accurate system measurement. Figure 5-4(c) and (d) show the reward surface and the integrated surface of the system outputs over the map. The similarity of the reward surface and the integrated output surface can be seen clearly through the contour plot of the two. In the contour plot of the integrated surface, the generalisation of the system in the flat surface appears to be a reasonable expansion of the reward slope.

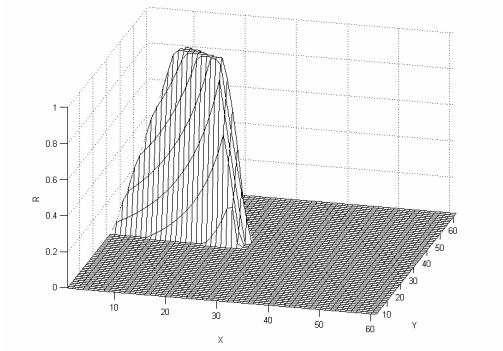
[1] The integrating function is an inverse of gradient function in MATLAB and the source code locates at http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=9734&objectType=scoreDetails.



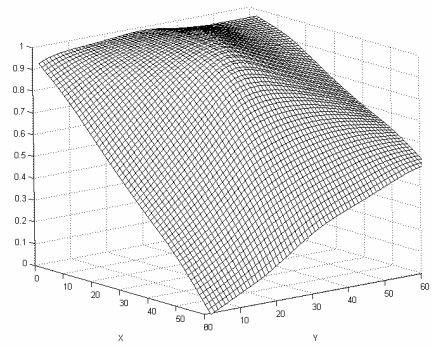
(a)The contour simulated reward surface



(b)The contour of integrated system output surface



(c) The reward surface



(d) The integrated system output surface

Figure 5-4 The reward surface underlined by the learning system after 30 walks with a comparison to the reward surface that represents the human's feedback.

However, the surface plots of the two, Figure 5-4(c) and (d), appear to give a visual difference. This is only because the generalised flat surface area in the integrated output surface has such a steep gradient and the gradient in the original reward slope area is too small to show a contrast. The integrated output surface provides no flat area and stretches the slope of [0, 1] smoothly over the whole map. In Figure 5-4(b), it can be seen that the slope is similar to the reward slope in the integrated surface within the contour line 0.98.

Figure 5-5 is the integrated surface above the plane: z=0.98, which shows the integrated surface near the area of the original reward slope. A strong similarity with the original surface is then illustrated. Generally speaking, the integrated system output surface is clearly a smooth surface with the same trend of gradient and similar cross sections as the reward surface.

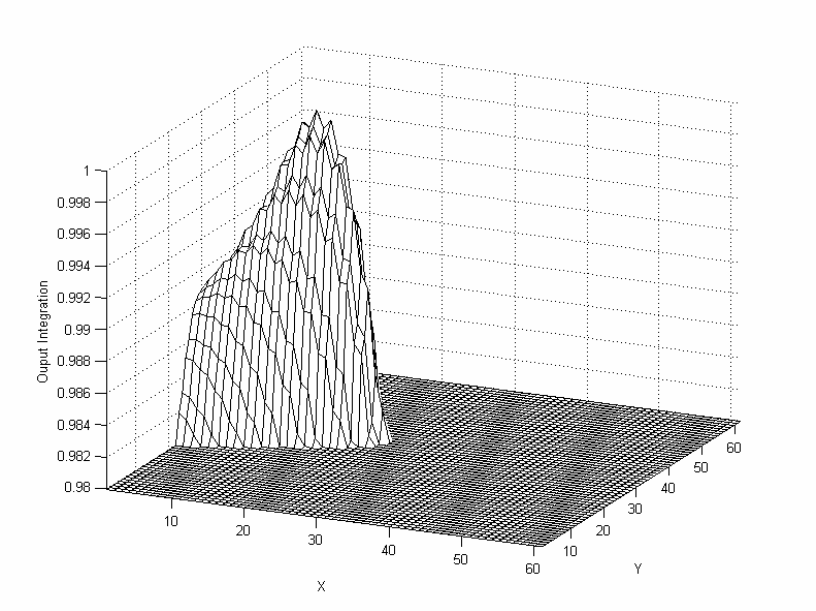


Figure 5-5 The integrated output surface near the original reward slope area, which has a similar shape to the reward surface with a different scale of gradient.

5.4.2 System Analysis

For an online system, it is necessary to observe its performance during operation. The widely used criterion of error performance will be superficial for the proposed system because the training data are collected and improved online as the learning progresses. Thus the errors are measured under different targets at different times. A set of observations therefore must be used to give a full view of the system operation. The following observations will be used in the discussion in this section:

System outputs

A successful system is expected to maintain zero outputs at the goal. These reflect the system movements and its stability;

Reward feedback

A learnt system should have a fast increase of the reward and the ability to maintain the top reward score. These provide an indicator of robot position as well as the quality of the learning;

Mean of Squared Errors (MSE)

As the training data are updated during the operation of the system, the value of MSE here thus should not be interpreted as a system performance measurement but

the change of MSE can still be helpful to explain system behaviours. A steep change of MSE reflects a big adjustment of the system weights and a successful system is still expected to have a fast descent and a good convergence of MSE. This is the most commonly used system measurement;

Training data updates

These give information about the size of the training data set as well as the behaviour of training data selection.

Appendix I has the complete data of these measurements of the simulations in this chapter. For a clearer view of the operation, we will selectively choose the most representative parts from the time axis for analysis. The early walks are worth careful analysis because the system is building the performance from scratch. Figure 5-6 shows the observation in the first five walks. In Figure 5-6, it can be seen that the first walk ended before the system stabilised and the reward feedback didn't reach the top score. As discussed in the previous section, this is because all walks are regulated to 400 steps for a clearer comparison. It can be seen that at the end of the first walk, the system output has the trend of stabilising and the reward feedback is clearly rising.

The reward history in Figure 5-6, shows that the first walk experiences decreases of the reward feedback at the beginning, which means the robot was going away from the goal and the training data collection is therefore working on the secondary rule as shown in Equation (5-7). But the system manages to steer into the right direction after learning for a few steps and the reward value rises as the robot approaches the goal. The system performance after the second walk of the system has been successful because all walks stabilise, and the reward stays at the top value. This means that the system only used one walk to build a weight basis that provides a correct general direction for the robot. Learning is then supported by new data. The fast learning of the system is again demonstrated. In the MSE figure, there is a faster descent of the error in the first walk than in other walks. The system demonstrates the ability to learn fast during early operations.

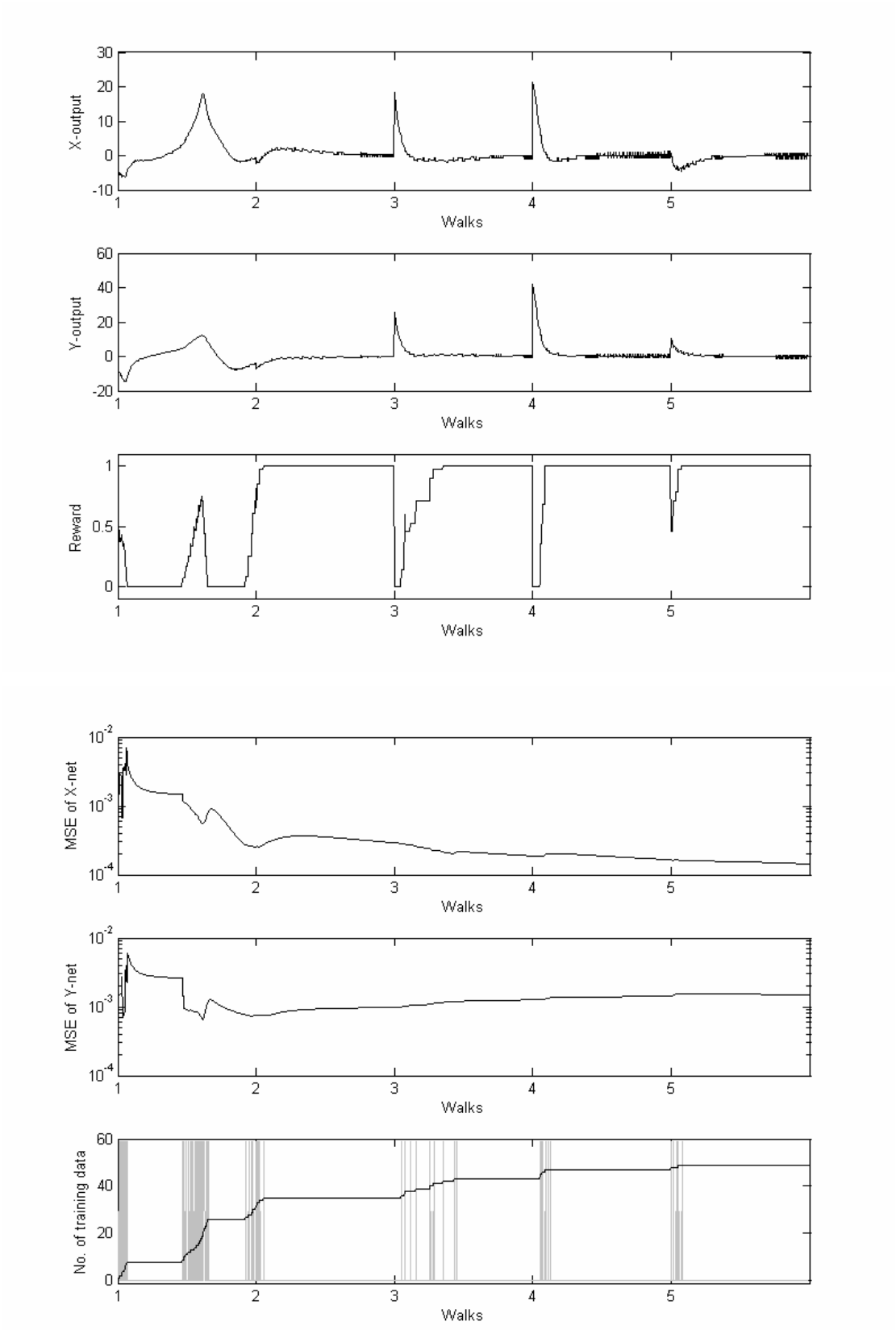


Figure 5-6 The data and measurements of the first five walks of the system based on the reward surface with one reward peak and a large area of flat surface.

In MSE, although every measurement is made from different targets as the training data are updating, the general trend of fast descending still demonstrates the system's ability to adapt. The fact that the error measurements keep dropping even when the new training data arrive proves that the system has been altered in the early walks to match the feature of the reward surface.

In the plot of training data in Figure 5-6, grey bars indicate the time indices at which training data are updated. It can be seen that the number of training data values rises every time the training data set is updated, which means the system is collecting new data throughout the first five walks. If the training data are updated and the reward feedback is falling, the system works on the secondary rule of training data selection as Equation (5-7). Otherwise, the training data selection is working with the rule presented by Equation (5-6). It can be seen that, in the first walk, the steep change of the MSE is associated with a large number of training data updates, where a wide grey bar and a steep slope in the size of training data set can be found. To collect a sufficient number of training data patterns is important for the system building in the early walks. It is shown in Figure 5-6 that the first walks updated training data more times than the later walks. Comparing it with the reward feedback figure, it can be seen that about half of the training data are collected through Equation (5-7) and the other half through Equation (5-6). It is then evident that the learning of the system in the early stages has been supported by both rules proposed in training data selection. The proposed training data selection method therefore appears to be effective. When the system progresses to later walks, the reward doesn't drop anymore and the training data are then selected by Equation (5-6).

For an online system, the long-term system behaviour is also worth investigating. Although the system has appeared to be learning fast and successfully in the early operations, it doesn't ensure the system's stability in long term. Figure 5-7 shows the system performance during the 30^{th} to 40^{th} walks.

In Figure 5-7, the output and the reward history of the system show that the system performs well. However, it is noticeable that both systems appear to have a small oscillation while trying to stabilise. The oscillation exists because, rather than map the reward peak as a small flat area as it is in the reward surface, the neural networks recognise the peak as one single point in the map, which can be seen clearly in the integration of the output surface in Figure 5-5.

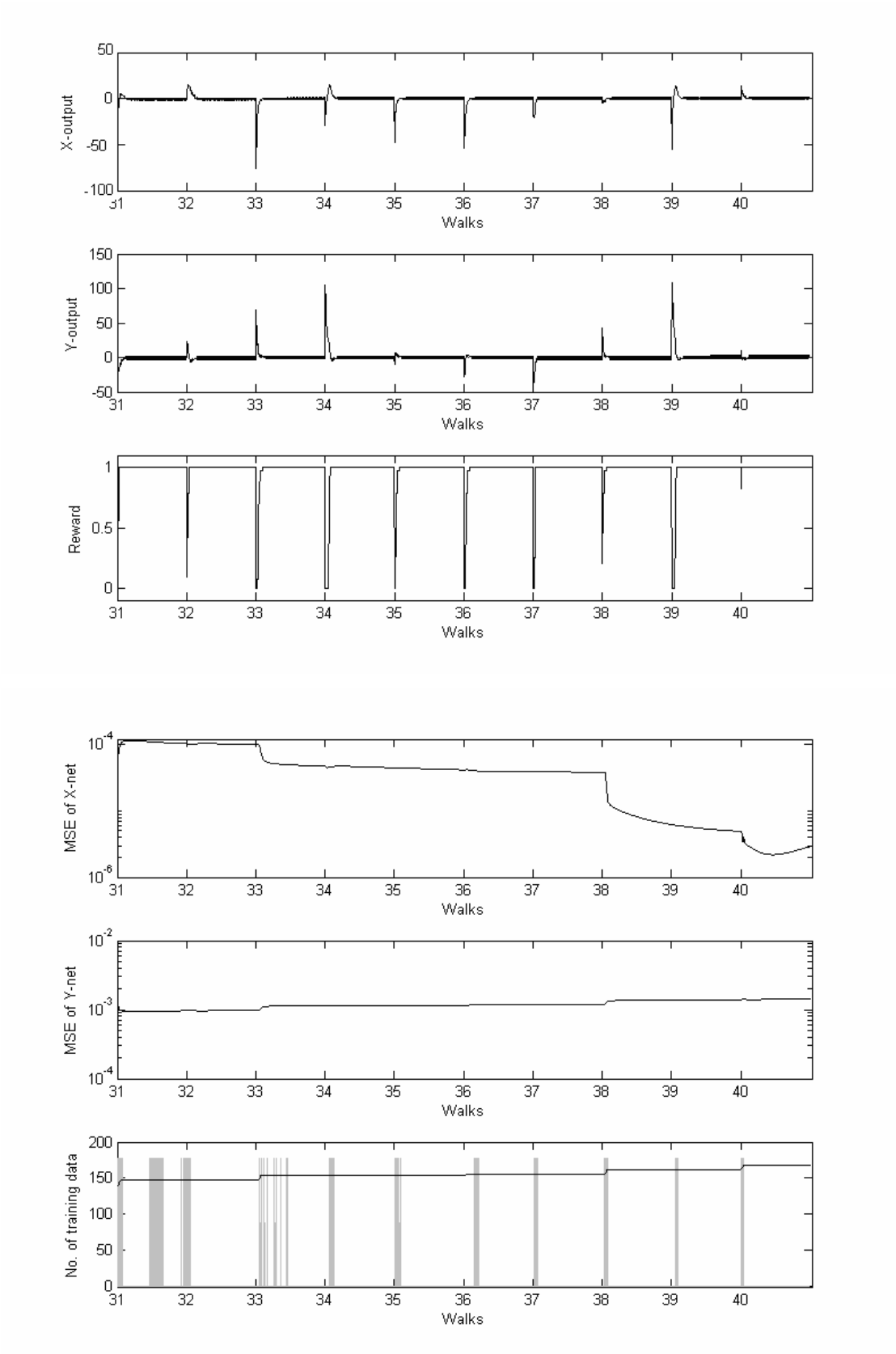


Figure 5-7 The data and measurements of the 30th to 40th walk of the system based on the reward surface with one reward peak and a large area of flat surface

The peak of the reward surface is actually another flat surface providing no training data and the recognition of the peak is based on the system's generalisation. In practice, it is rare for the robot to stop at the exact peak point for the system and thus the robot normally oscillates around it in the simulation. The magnitude of the oscillation is less than 5mm, which is 1% of the output limit. It is a high frequency and low amplitude oscillation that is unlikely to have observable effects on the robot motor system.

In the training data history, there are times that the number of training data remains the same but its contents are updated. These are the efforts of the training data collection method to replace the old training data with better-rewarded moves. It can be noticed that if the reward feedback rises, the training data are updated. In the selected reward surface here, the reward rises when the robot steps into the reward slope. It is expected that the entry is through a different point with possibly slightly different movements every time. Therefore, there can always be some data worth being updated on entry to the reward slope.

The MSE of Y-network is clearly stabilising. The MSE changes in X-network are at a very low value scale that can only trigger small weight adjustments. The difference between the two MSEs indicates that X-network is under more pressure to learn than the Y-network. This is because the reward slope is longer in width than height, which gives more responsibility to X-network to provide a correct recognition of the reward slope. By separating the movements of two directions into two networks, one network avoids compromising the performance of the other. The use of two networks has therefore demonstrated an advantage.

Although the MSE of X-network at the 40^{th} walk appears to be rising, introduced by the new data collected at the start of the walk, it stabilises later with a value less than 10^{-4}, the details of which can be found in Appendix I. The MSE of the Y-network maintains a value of approximately 10^{-3}. As it has been mentioned that only the stabilisation of convergence is the interest of investigation here and the exact value of the convergence is not an appropriate performance measurement. In this simulation, the system has demonstrated a very promising ability of using-while-learning in the scenario of locating the appropriate position to follow a person.

5.5 The Capabilities of the System

The system studied in this chapter uses multilayered feedforward neural networks with backpropagation. Two networks have been employed, each of which is responsible for one Cartesian axis in the 2-dimensional human reference frame. The training is based on the error backpropagation proposed by Hecht-Nielsen (1989). Because the model provides no prior training data, an online training data collection method is proposed. The error measurement function is thus based on the training data collected and the learning rate is adjusted dynamically, based on the error measurements.

Compared to the learning speed of over 2000 walks in the reinforcement network studied in Chapter 4, the backpropagation learning has shown distinctive advantages. It learns much faster than the reinforcement network, where only one walk has been used to produce a generally correct direction map. After about 40 walks, the system has generated the whole map well, providing better paths than the reinforcement network after 2000 walks. The system outputs have shown the advantage of being able to generalise with limited training data. The analysis of the system with multiple measurements has shown a plausible using-while-learning system behaviour.

However, the results in this chapter are only based on the reward surface with a simple feature, one reward peak. It is unreasonable to conclude that the human preference will always be like this, as has been discussed in Chapter 3. It is sensible to believe under certain scenarios or with certain individuals, the reward surface can be more complex. In the next chapter, the study will investigate the system behaviour in a reward system with more complex features. It will target potential issues that still need to be considered regarding the complexity of the interaction. The study will propose the idea of adaptive and reactive learning, based on which a novel improvement will be introduced: a multilayered feedforward system trained with two learning threads.

Chapter 6 Online Backpropagation with Two Learning Threads: Adaptive and Reactive Learning

Chapter 5 studied one possible solution to the problem of online learning to find a preferred spatial location via human feedback. This process used a multilayered feedforward network trained with error backpropagation. Simulations have demonstrated the system's ability to learn fast during online operation as well as to generate reasonable policy to control the movement of the robot. The ability to generalise removes the need for lengthy pre-training existing in reinforcement learning studied in Chapter 4, and the fast learning speed secures the system's acceptable performance in the initial learning stage. The simulation in Chapter 5 has demonstrated the system's ability to function regardless of the degree of learning as well as the ability to improve the performance as the operation progresses. In the simulation scenario used in Chapter 5, the system is using-while-learning.

However, the results in the previous chapter only supported the system's ability to work with a simple reward surface, limiting human preference to a simple reward slope. It is not convincing that human preference will always be such a simple map. In this chapter, therefore, the study will investigate the system's performance with a complex reward surface, target the issues revealed by the simulation results, and raise the discussion of two types of learning threads: adaptive and reactive learning. It will then propose a novel modification of the backpropagation learning strategy, which will increase the flexibility of the online learning system and enable the system to operate smoothly with complex human preferences. Further simulations will then be presented to support the using-while-learning ability of the updated social positioning behaviour when faced with a complex reward surface.

This chapter consists of 5 sections. Section 6.1 explains the complexities of learning that haven't been discussed in Chapter 5, and introduces one scenario that involves such complexities. The system studied in Chapter 5 is re-tested in the proposed new scenario and the analysis concludes that the system needs

improvement. The concepts of adaptive and reactive learning are then introduced. Section 6.2 proposes the learning algorithm of reactive learning. Section 6.3 identifies and analyses the effects of adaptive and reactive learning by short comparative simulations, and proposes the system that is trained with both types of learning. Section 6.4 tests the modified system in the scenario introduced in Section 6.1 and the analysis supports the system's ability of using-while-learning in a more complex scenario. Section 6.5 presents our interactive simulations that allow the human to give feedback to the system, where the learning system faces a more difficult and dynamic situation than other simulations. Finally Section 6.6 concludes our findings in this chapter and summarise the capabilities of the updated social positioning behaviour.

6.1 Analysis with a Complex Reward Surface

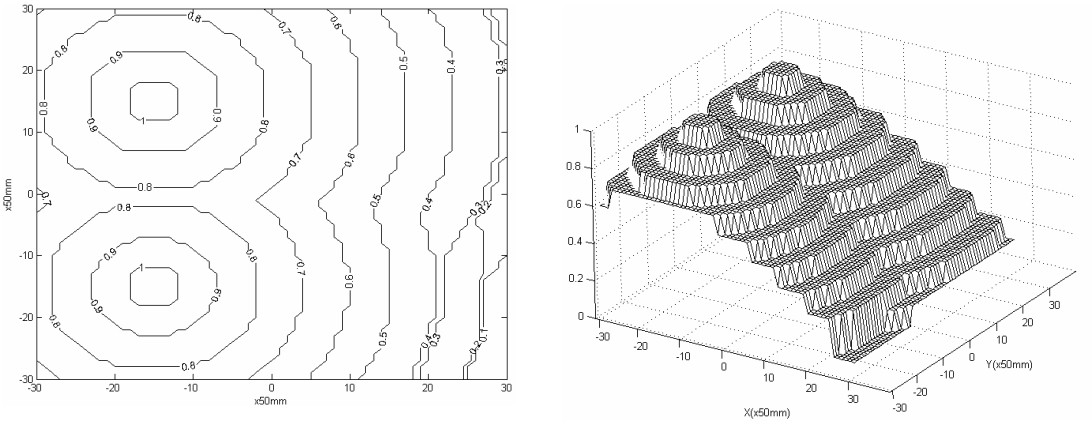
The simple reward surface that has been studied in the previous learning systems only has one reward slope. A complex reward surface however may contain multiple reward slopes to reflect the complexity of a human mind or the environment. Because the learning is explorative, meaning that the robot only learns based on what it has experienced, a constant complex reward map can introduce the issue of learning new features during operation. Although it has been assumed that the reward surface remains consistent in each individual, there are still situations in which the human provides new information about the reward surface, which has not previously been learnt by the robot.

6.1.1 The Complex Learning Scenario

When the human's preference has multiple features, there is a good chance that the movement of the robot will be regionally clustered. Therefore the system function generalised by the positioning behaviour can only reflect the features of the human's preference in the limited region that the robot has experienced. If the robot is then exposed to a new part of the map, the system may have to readjust its function if the human preference in the new region doesn't match the system function generalised by the data collected in previous region. When the human's preference in the new region is totally different, this adjustment can be as large as learning something totally new. The following discussion in this section aims to analyse the system's performance when facing the challenge of learning new features online while the system is also

required not to alter its learnt useful knowledge and not to lose the ability of using-while-learning

Considering a normal office corridor with a width of about 2 meters, it can be easily observed that people usually walk on one side of the corridor. If someone walks with his right side against the wall, he can only ask the robot to follow at some point in the upper plane of the human coordinate frame (see Figure 6-1(a)). When the human is walking with his left side against the wall, the robot can only move in the lower plane and can't move into the peak mapped by the previous walks. The human has to pick another position on his right hand side for the follower. In this situation, the reward surface of the human has two reward peaks. The human can only direct the robot to one of the two, the one that locates on the current half of the plane in which the robot is moving. Figure 6-1 gives an example of the human's reward surface in such a situation. This reward surface will be used throughout Sections 6.1 to 6.4.



(a) the contour plot of the reward surface (b) the surface view of the human preference

Figure 6-1 Reward surface with two reward peaks. The width (Y-axis) of the reward map is assumed to be twice the width of the corridor. When the human is walking at one side of the corridor, his actual preference is the corresponding half of this reward surface.

After the system has been used on one side of the map for a period of time, it will be well developed to recognise one peak of the map. Our research is interested in how the system responds when the human suddenly walks on the other side of the corridor and places the robot on the other half of the map. When the system operates on one side of the map, no training data pattern exists to reflect the feature of the other half of the reward surface. The output over the other half of the surface is therefore based on the generalisation of the backpropagation. When the robot is placed on the other half of the map, the training data collected based on the human's feedback directs the robot to the other peak that has not been learnt by the positioning

behaviour. The robot then needs to work against its previous knowledge and to change the system to recognise the new peak in the reward surface. This adjustment of the system is also expected to be quick enough to be a part of the using-while-learning operation.

Compared to the reward surface used in Chapter 5, in addition to the number of peaks, the flat areas in Figure 6-1 are also organised differently. Figure 6-1 has no smooth reward slope but sparse gradients. Rather than one big flat area, Figure 6-1 has many narrow flat areas spreading through out the map. This means there is no longer a need to restrict the starting point of the robot. But it also means that the system can't have as many training data patterns as there are at the beginning of the learning in Chapter 5. This is another challenge faced by the learning system.

6.1.2 Simulation of Learning a New Feature

A simulation was formed in the situation described in previous section. The system was first simulated with only upper half of the map available. At first, 40 walks were performed in this situation as this is about the time length for the proposed learning system to develop a mature performance, based on our observation in the previous chapter. Then the robot was placed in the lower half of the map for the next 20 walks. Each walk was still limited to be 400 steps and the starting points of the robot were selected randomly within the assigned half of the map. The full data of this simulation can be found in Appendix II. Figure 6-2 gives some examples of the walks during this operation.

In Figure 6-2, we can see that the first walk failed because the robot went out of the map, in which case it could have hit the wall in practice. The simulation continued anyway and the system managed to turn the robot around but it still didn't manage to steer the robot into the reward peak. This shows that collecting fewer training data patterns during early learning has clearly added more pressure for the system to remain using-while-learning. Nevertheless, this improved as the operation progressed and more data were collected. Further walks shown in Fig 6-2 at the upper map are successful. The walks in the lower half of the map are the history of the robot movements in the later 20 walks. The 41st and 42nd walks were the first two walks in the later 20 walks. Neither walk achieved the goal. We can see that the speed of learning had decreased while the system attempted to learn new features. Later walks,

the 52nd and 58th one, were struggling at the local minimum outside the desired peak. The system obviously didn't reach the desired performance during the later 20 walks.

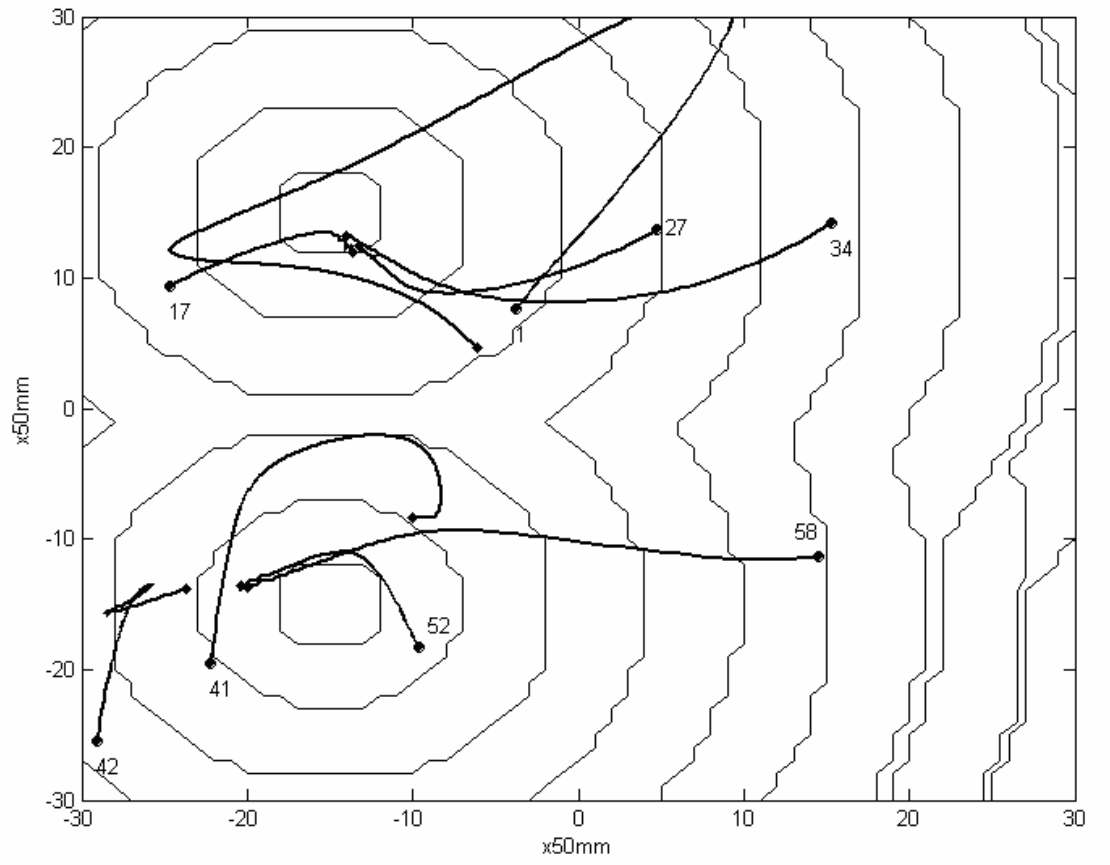
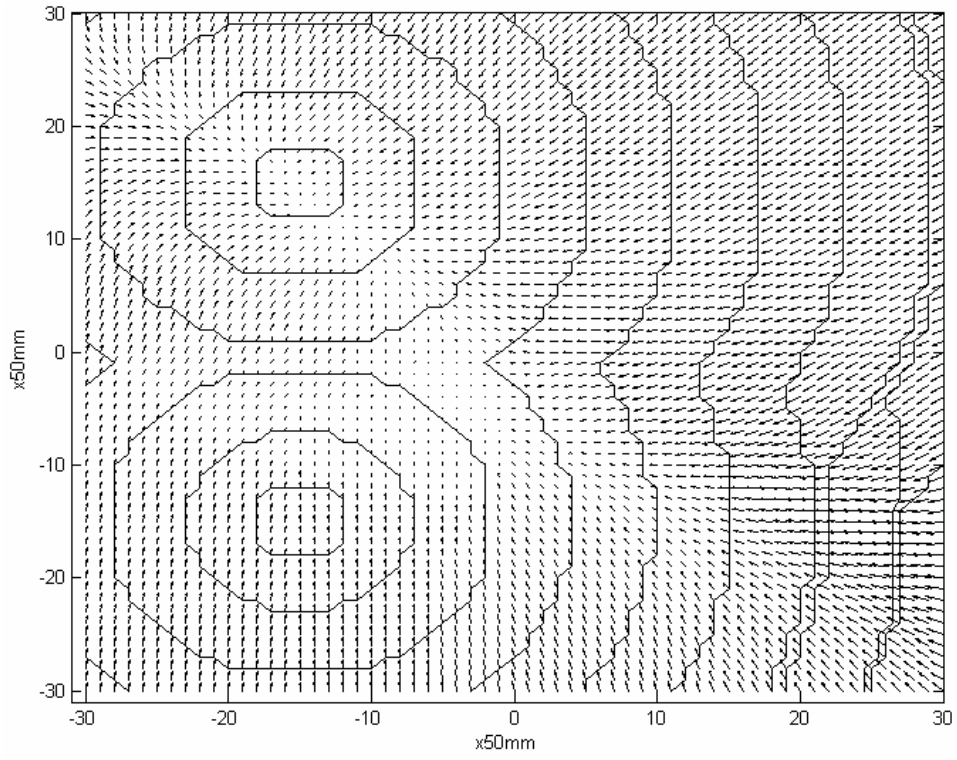
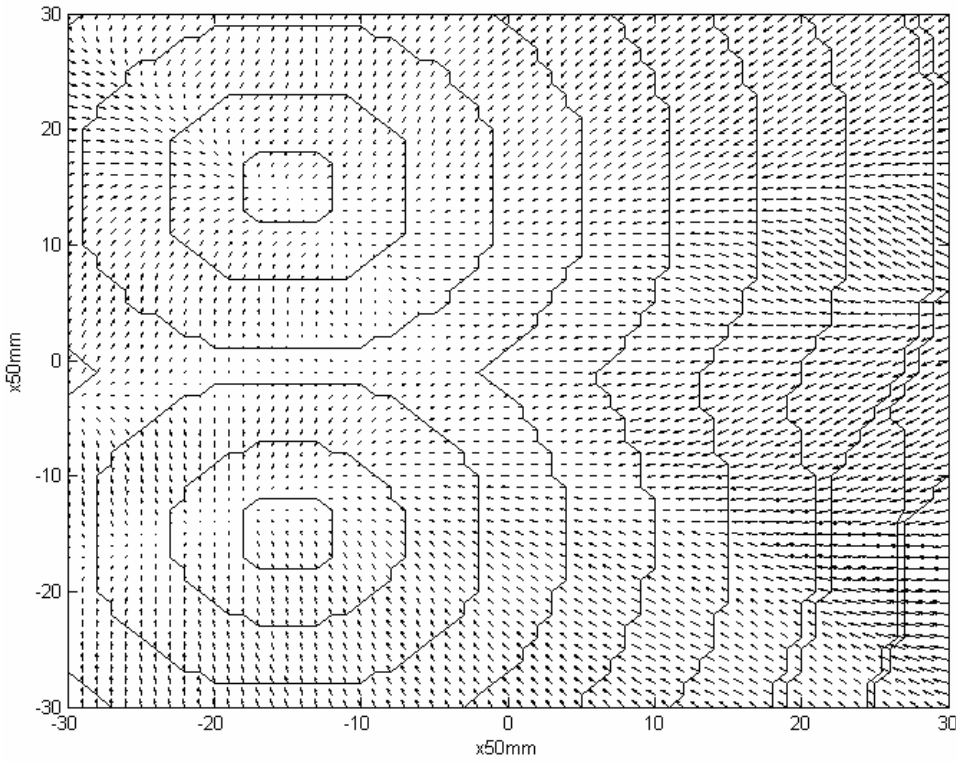


Figure 6-2 Some walks of the system simulation with backpropagation learning (as in Chapter 5) and two reward peaks. The ● marks the starting point of the robot and ♦ marks the ending/stabilising point of the robot. The numbers indicate the index of the walk in the operation.

Figure 6-3 shows the system output surface after the 40th and 60th walks. From the integrated output surface in Figure 6-4(a), it can be seen that other half of the map, where the robot has no access, has a generated output surface directing the robot to the peak that the robot has experienced. For convenience of reference, in the remainder of this chapter, this peak will be referred to as the *upper peak* of the reward surface shown in Figure 6-1. The other reward peak will be referred to as the *lower peak*. In Figure 6-4(b) we can see that the system has changed to create the lower peak but from Figure 6-3(b) we know that the location of the lower peak is not correct.

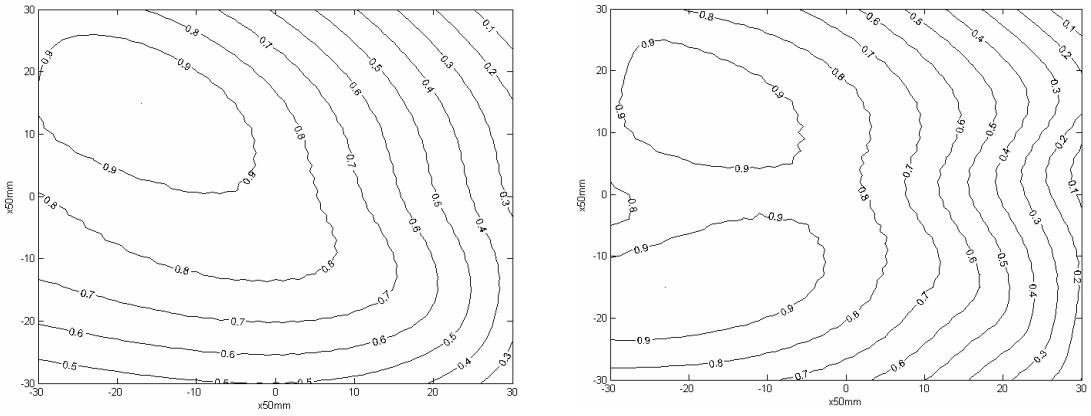


(a) System output surface after 40 walks



(b) System output surface after 60 walks

Figure 6-3 The system output surface after 40th walk and 60th walk in the simulation with backpropagation learning (as in Chapter 5) and two reward peaks.



(a) Integrated system output after 40 walks (b) Integrated system output after 60 walks

Figure 6-4 The contour plot of the integrated system output surface after 40th walk and 60th walk in the simulation with backpropagation learning (as in Chapter 5) and two reward peaks.

Figure 6-5 contains the system measurements during the later 20 walks. In order to have a comparative view of the impact of the change of the environment, the data in Figure 6-5 start with the 39th walk, the last walk on the upper half of the reward surface. This walk has stabilised outputs as well as converged error performance. The record from the reward history shows that it was a successful walk. It can be seen that in the following two walks, the system had a sudden increase in the system error performance. This was because the new training data didn't match the output surface of the system, which indicated that the robot was experiencing a situation that was different from the learnt information.

A much larger oscillation is shown in the Y-output than X-output. This is because the position of the new peak is different from the learnt peak only in its y coordinate. The system appears to have difficulty in stabilising its performance on the new reward peak, the lower peak. The reward history shows that the system experienced a number of failures and that it eventually formed a local minimum near the lower peak as the robot converged at the position with a reward value of 0.9 rather than 1. It is clear that the system proposed in Chapter 5 lacks the ability to react to changes in the reward surface. In a using-while-learning online system, this can cause a set of walks to fail during the middle of the operation. Even if the human preference offers a certain degree of consistency, it is still important for a social learning system to be equipped with the ability to react to sudden changes in the environment.

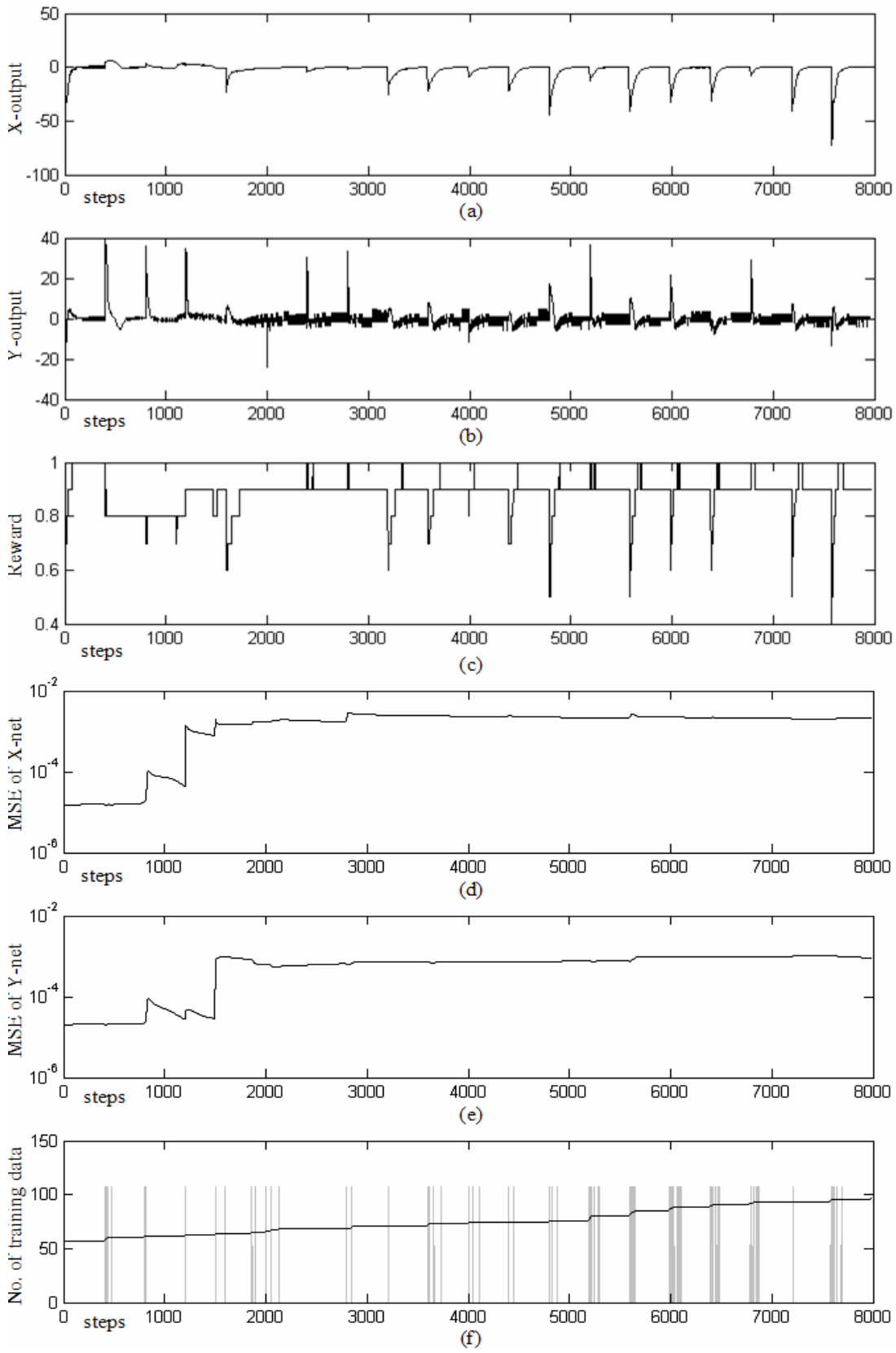


Figure 6-5 The system measurements for the later 21 walks in the simulation with backpropagation learning (as in Chapter 5) and two reward peaks. The first walk is the last walk on the upper half of the reward surface.

6.1.3 Adaptive Learning

The classic backpropagation algorithm proposed by Hecht-Nielsen (1989), normally has a slow convergence. The speed of learning for backpropagation has been widely discussed in the literature (e.g. Battiti, 1992, Hsin et al., 1992, Ahmed et al., 1995, Chen, 1990). This, however, can't be taken out of the context of the nature of training introduced by those research projects. It has to be noticed that those studies are mostly based on offline learning by the network. Backpropagation in multilayered feedforward networks usually has a large set of training data to describe the features of the target function. Under such a scenario, the systems with classic backpropagation normally experience a very low rate of convergence. Many proposals have then been recognised to be helpful for offline learning using backpropagation (e.g Foresee and Hagan, 1997, Naimin Zhang et al., 2006). However, when the network is explorative and is working online with the proposed online training data collection, the situation is different.

In the proposed system, it is obvious that when the system starts to collect training data it can at most collect one pattern of training data at each step. The number of training data patterns increases through time. The initial training of the system therefore is only based on a very small number of training data patterns. The neural network approximates the target function by finding the best fit among all training data in a least square sense. Because the learning starts with a few patterns of training data, the best fit among the data is easy for the system to find and the decrease in error at an early stage is therefore guaranteed to be fast. Further collected training data support the learning outcome of the early stage, so the system only has small adjustments to make while the initial decrease appears to be stabilising. This reveals why the system can work in a using-while-learning basis in Chapter 5 using classic backpropagation learning.

The system error measurement of the simulation increased in the previous section when the system was exposed to the new feature of the reward surface, at which point the system was required to find a new best fit among a larger set of data. The set consists of a large amount of old training data supporting the upper peak and a few new training data patterns supporting the lower peak. This is the reason that the forming of the new lower reward peak is not as quick as the forming of the upper reward peak at the early stage of learning. In the simulation in Chapter 5, the system

develops a correct foundation when the scale of the training data set is small. When the set gets larger, only minor tuning is needed. However, when the robot faces new features after the system has been running for a while, the system has to re-adjust itself based on the large set of training data. The weakness of backpropagation during offline learning then has similar effects on the online learning here as well. It has a slow rate of convergence.

It can be seen in the error measurements in Figure 6-5(d) and Figure 6-5(e) that the system error decreases slowly. When the robot faces the new feature, the number of new training data patterns is very small compared to the existing training data. Therefore only a small part of the weight adjustment is based on the existence of the new peak. Hence the error falls slowly and the forming of the new lower peak is difficult.

One of the possible directions is to increase the learning rate to enhance the convergence of the system. However, this action is risky. A large learning rate and large system adjustment destroys the system's performance on generalisation. Foresee (1997) points out in his research of backpropagation learning that continuous steep adjustments of weights can eventually lead to an unsmooth output surface and a failed system generalisation. A small learning rate also contributes to the convergence of the system. Although the rate of converge can be slow, the system can converge to a good level of error and can stabilise well. The focus of the learning with small weight adjustments is *adaptation* that, in this thesis, refers to the objective of providing a smooth and well generalised surface based on all the training data collected through time. We thus refer the learning method used in Chapter 5 as *adaptive learning* in this thesis. This system relies on successful generalisation from the adaptive learning to complete the walk successfully and the walk fails otherwise. It is then unwise to simply increase the learning rate of the adaptive learning.

Rapid adjustment of weights is only needed in this system when the system is facing new features. Any new training data that may introduce a new feature must be the most recent experience of the robot. Therefore, a new focus of learning can be set to give higher adjustment based on the most recent training data. This is equivalent to making the system learn on a local error descent basis, meaning that the training data of this learning thread will only imply a local piece of the error surface. The aim of the learning is then to force the system make steeper adjustments based on recent data so that, if any features are new, the system reacts to them quickly and successfully,

even when a successful generalisation of the whole map has not yet been achieved. However such steep adjustments should only be applied during a short time period when there are new features, in order to avoiding risking the system's generalisation. Compared to the adaptive learning, the learning thread with the focus on newly selected training data is referred to as *reactive learning* in this thesis.

6.2 Reactive learning

The objective of reactive learning is to enable the system to respond to the human feedback faster so that the robot walks in the right direction by following the recent feedback without relying on the generalisation of the whole map. In such a way, the system reacts to a new feature of the environment quickly and lowers the risk of failing walks before the system can successfully recognise all the features described in the, possibly large, training data set collected online.

In order to achieve high reactivity, the system needs to adjust itself quickly to the local circumstances, namely the local gradient of the error surface. The local gradient is reflected by the sum of system error performance on the most recent training data selected from the current path, through which the robot is moving. These data only cover a short time slice from the present. Thus at any time k, the training data set for reactive learning is $\mathbf{T}_r = \{[\hat{\mathbf{p}}^t, \mu_t(\hat{\mathbf{p}}^t)] : k-c \le t \le k\}$, where c is the length of the selected time window, $\mu_t(.)$ is the training data memory as introduced in Section 5.2, and $\hat{\mathbf{p}}$ is the closest state to the input vector, $\mathbf{p}$, in the memory. The error performance function used for training is the weighted squared error:

$$\mathbf{E}_r^t = \sum_{k-c\le t\le k} \frac{1}{2}\left[\frac{\mu_t(\hat{\mathbf{p}}^t) - \tilde{\mu}(\mathbf{p}^t)}{K}\right]\cdot\left[\frac{\mu_t(\hat{\mathbf{p}}^t) - \tilde{\mu}(\mathbf{p}^t)}{K}\right] e^{-\alpha(k-c-t)}, \quad (6\text{-}1)$$

where α is a constant that is 0.2 throughout the simulation and the value is empirically justified, $\tilde{\mu}(.)$ is the system function and $\mathbf{p}$ is the system input vector.

The learning minimises the errors of the local training data $\mathbf{E}_r^t$. This learning is fast in that the number of training data patterns engaged is small and an even faster speed of learning can be achieved by using a larger learning rate, compared to the adaptive learning. It is still important to adjust the learning rate, η_r, as in Section 5.3.2:

$$\eta_r = H_r(1 - e^{-\beta\sqrt{E_r}}) \quad , \quad (6\text{-}2)$$

where $H_r \in \Re^+$ is the learning rate limit and β is a constant specifying the speed of the descent of the learning rate relative to the decrease in system errors. The value of β is 0.2 throughout the simulations in this thesis so that high error measurements will be assigned with a much larger learning rate than low error measurements. H_r is 0.5 throughout the simulations.

The next step of our study in the following sections is to further identify the possible roles of each learning thread and to propose a possible way to combine the benefits of the two types of learning.

6.3 The Effects of Reacting and the Combination of Learning

The purpose of this section is to identify the impact of reactive learning on the system as well as to propose and justify a way of combining reactive and adaptive learning. The methodology of this section is to test the system's performance with different learning threads from identical initial conditions and to compare the results of the simulations. Three simulations will be demonstrated in the reminder of this section: a system with adaptive learning, a system with reactive learning and a system with both learning threads. The way of training the system with both learning methods together will be discussed later in Section 6.3.3. All simulations use neural networks with the same initial weights, which are the system from Section 6.1 after its 40th walk and have been trained to recognise the upper peak of the reward surface only. The corresponding training data set is used to initialise the training data set of the simulations. The system output surface of the initial networks is as shown in Figure 6-3(a).

The chosen start position for the simulations is [-700, -1300]mm, which is [-14,-26] on the map shown in the following of this section. This position is chosen because it is close to lower peaks but it is on a direct path towards the upper peak based on previous learning. Thus the system has to respond fast to the feedback to recognise the lower peak. Because the initial movements of the robot are expected to follow the initial system output surface and go towards the upper peak passing through the lower peak, it is guaranteed that some training data indicating the

existence of the lower reward peak can be collected. Each simulation takes one walk, i.e. 400 steps.

6.3.1 System A: Adaptive System

The simulation results of the system using adaptive learning only in the environment stated above are shown in Figure 6-6 and Figure 6-7. It can be seen that this walk failed as the robot couldn't recognise the lower peak in time. The route shown in Figure 6-6 doesn't demonstrate obvious efforts by the system to adjust to the lower peak.

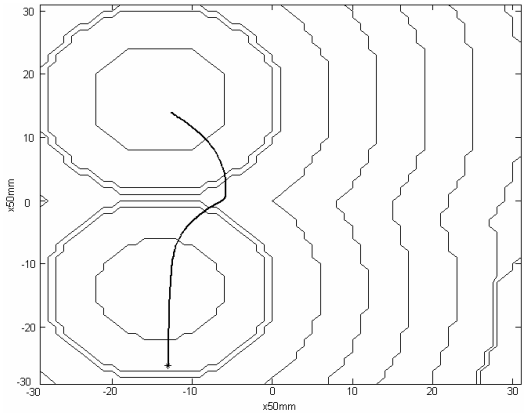


Figure 6-6 The walk produced by System A. The system uses adaptive learning only. The system is facing the new lower reward peak for the first time with the knowledge of the upper pre-learnt from 40 previous walks. * marks the starting point.

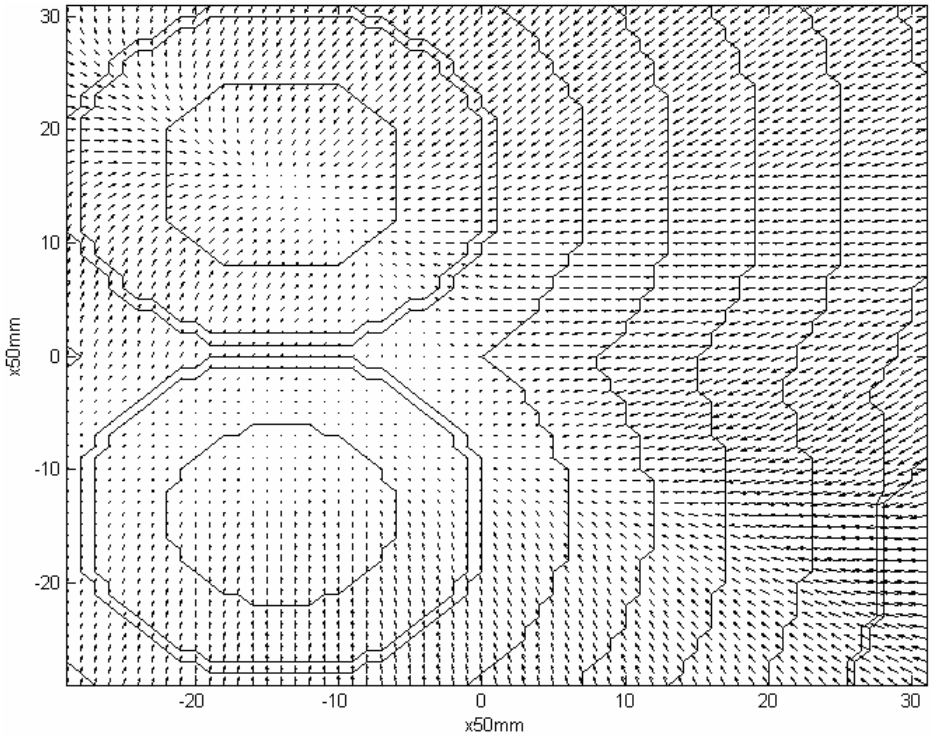


Figure 6-7 System output surface of System A after the walk. The system uses adaptive learning only. The system is facing the new lower reward peak for the first time with the knowledge of the pre-learnt upper peak from 40 previous walks.

In the output surface after this walk, as shown in Figure 6-7, it can be seen that the system is in the process of forming a new peak as a minimum area appears right above the lower peak. However this process was not acting fast enough and the walk therefore failed. Nevertheless, the advantage of adaptive learning is also demonstrated. While trying to map the new feature in the environment, the part of the previous generalisation of the system, which has no conflict with the new features, has been well preserved. This enables the system to maintain the previous knowledge correctly while learning new features. Calculating the difference between the upper output surface of Figure 6-7 and Figure 6-3 (a), the MSE vector between the two surface is $[1.35\times10^{-5}, 4.9\times10^{-4}]$. We will use this measurement in the following sections to compare the system's ability to maintain the previously learnt knowledge, where a low MSE vector denotes a better ability to retain the learnt knowledge as it means the difference of the recognition of the upper reward surface of the current system and the initial system

6.3.2 System R: Reactive System

This simulation runs with the system using reactive learning only. Having mentioned that a system with a large learning rate and small coverage of the general features in the training data can be risky, the simulation results of the walk in Figure 6-8 demonstrate such risk as well as the advantage of reactive learning. The reactive learning windows length, *c*, is 10 steps and the system was trained pattern by pattern.

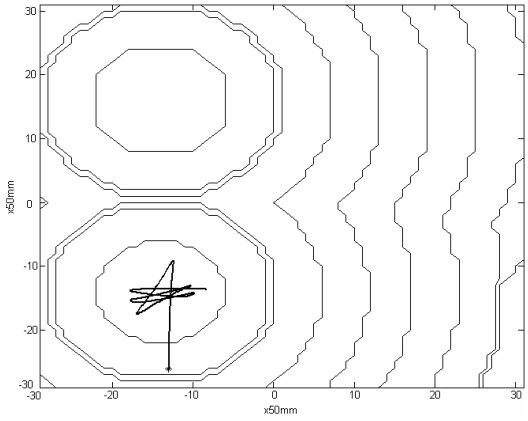


Figure 6-8 The walk produced by System R. The system uses reactive learning only. The system is facing the new lower reward peak for the first time with the knowledge of the pre-learnt upper peak from 40 previous walks. * marks the starting point.

The first turn point in the path of the robot indicated the first recognition of the new peak. The system then made an appropriate turn at the second point. However, the robot failed to stay at any position. The movements of the robot are always heading to the nearest minimum that the system forms. The oscillation thus shows that the system was constantly forming new minima without being able to converge to any of them. This is a common result and a risk of using a large learning rate. Although the general trend of the system adjustment can be correct and steep, it lacks the ability to converge and generalise a smooth surface. In the output surface shown in Figure 6-9, it can be seen that the reactive learning has corrupted the initial output surface much more than the adaptive system. The MSE vector between the upper half of Figure 6-9 and Figure 6-3(a) is [1.13×10^{-4}, 1.5×10^{-3}], which is about ten times larger than the results from System A in previous section. Although reactive learning has shown a very fast reaction to the new features of the environment, its destructive effects on the system generalisation is obvious. The learning didn't converge. The system oscillated with a large magnitude. Reactive learning lacks the ability to generalise and converge.

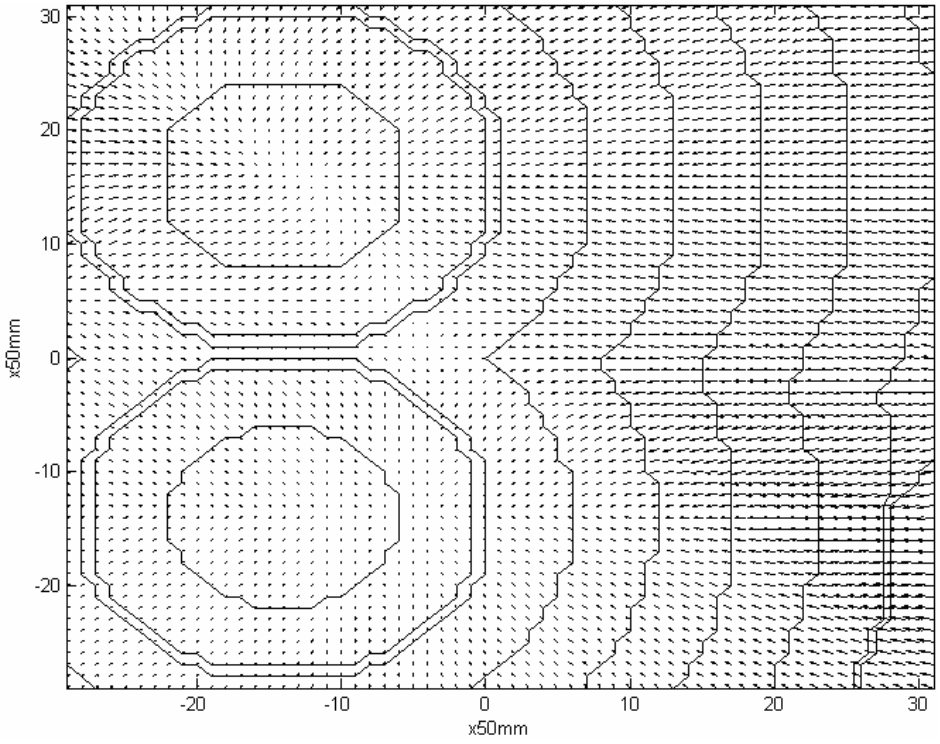


Figure 6-9 System output surface of System R after the walk. The system uses reactive learning only. The system is facing the new lower reward peak for the first time with the knowledge of the pre-learnt upper peak from 40 previous walks.

6.3.3 System A-R: System that is Both Adaptive and Reactive

Reactive learning is responsive but the large learning rate and training on local data generalise poorly in the scale of the whole map, on which the movement in the flat surface relies. When the network has shifted the learning into minimising the local error performance, it doesn't stabilise to any output surface and it doesn't optimise or generalise either. Therefore, it is necessary to combine the advantages of reactions and adaptations in such a way that they both contribute to the system to fulfil our objectives of using-while-learning.

6.3.3.1 The Training Process

Mixed learning exists in the literature. Some research projects, such as the research of Blanchard (2005), mainly focus on different learning rates at different time scales. However, it requires a timetable to specify the degree/type of learning. The proposed learning scenario, in this thesis, is online. The human preference and training data can change at any time, the network may have to continue to learn indefinitely or shut down the learning at any time. It is very hard to compose such a timetable. Other relevant applications from researchers such as Huang (2005) focus on hybrid learning with multiple criteria. They are mostly used to aid the system convergence by introducing additional cost criteria.

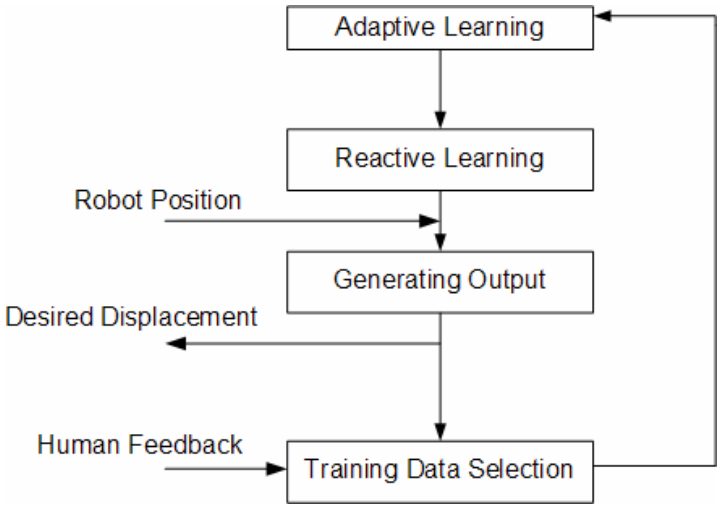


Figure 6-10 The routine of training the social positioning behaviour system, using adaptive, reactive learning threads and training data selection.

We propose a novel approach where two independent learning threads manipulate the same network together. Figure 6-10 suggests a possible way to combine the two learning threads in one network by training the system with the two

learning threads in order. Each learning thread runs one epoch at every system step. The training thus consists of three procedures: the adaptive learning, the reactive learning and the training data selection. They are organised as shown in Figure 6-10 in order to process the learning properly.

The proposed training routine in Figure 6-10 trains the network with both learning threads in turn. The order in which the two threads were organised didn't show any observable effects on the learning outcome in our tests. This routine is built for online learning. When the human feedback is received, the training data are selected where possible. Although the learning threads have been assigned to run in a certain order, because the learning rate of the two learning approaches are controlled by different error measurements, a dynamic shift of the control of the system is expected, more details of which are discussed later in Section 6.3.4.

6.3.3.2 The Simulation of One Walk

The walk of the system with both learning threads is shown in Figure 6-11. From the path of the robot, it is clear that the system recognised the new feature quickly and correctly.

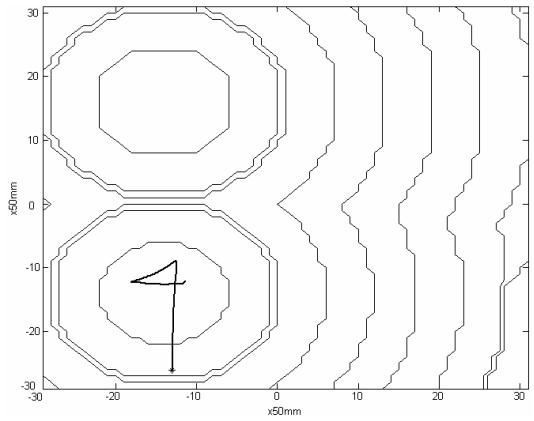


Figure 6-11 The walk produced by System A-R. The system uses both reactive and adaptive learning. The system is facing the new lower reward peak for the first time with the knowledge of the pre-learnt upper peak from 40 previous walks. * marks the starting point.

The new minimum of the system appeared when the robot was at the first turn point, similar to the reactive system. After a second correction at the second turn point of the path, the system found the new reward peak and the upper peak showed no obvious change. The output surface after the walk is shown in Figure 6-12.

The MSE vector between the upper half of initial output surface in Figure 6-3(a) and the same area of the system output here is $[1.35\times10^{-5}, 4.9\times10^{-4}]$. This MSE value is exactly the same as the MSE vector between the adaptive system

in Section 6.3.1 and the initial system studied. This can mean that the upper half of the output surface here is very close to the upper half given by System A: the adaptive system. As a matter of fact, the MSE vector between output of, System A, and this system at the upper half of the output surface is only [9.9×10^{-6}, 1.3×10^{-4}]. This means that the system has maintained the adaptive learning's generalisation while introducing the reactive learning thread. It is then obvious that, when the system is equipped with both adaptive and reactive learning threads, the system has the ability to react quickly to the new features of the environment as well as to converge and generalise well.

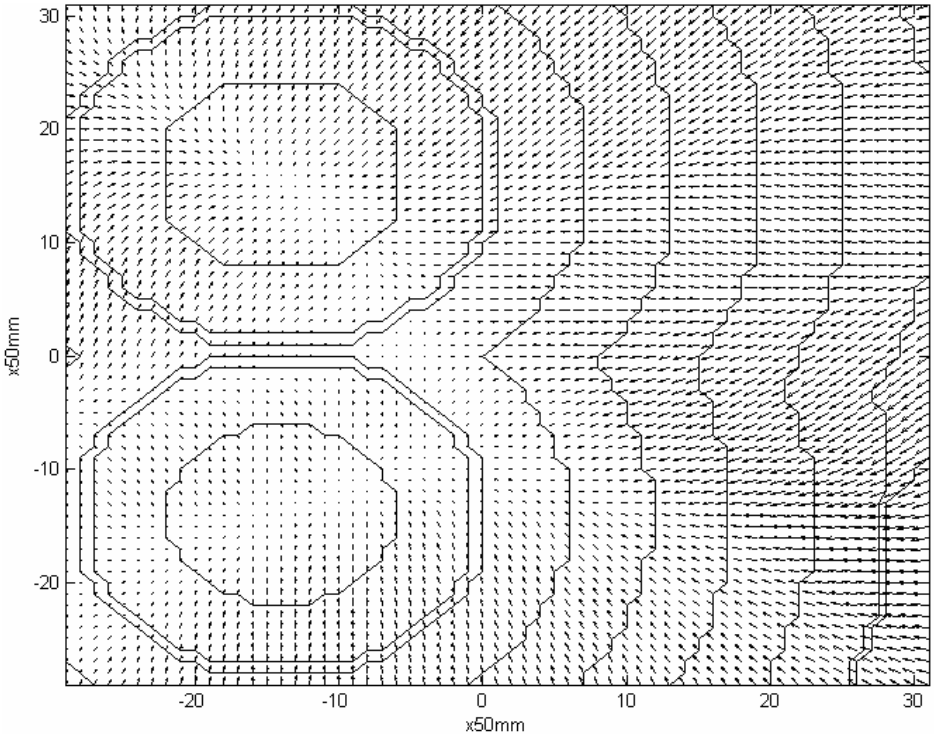


Figure 6-12 System output surface of System A-R after the walk. The system uses both reactive and adaptive learning. The system is facing the new lower reward peak for the first time with the knowledge the pre-learnt upper peak from 40 previous walks.

6.3.4 Comparative Analysis

In this part of Section 6.3, we will carry out a comparative analysis of the measurements of the three system simulations above. The purpose of the following analysis is to understand and contrast the different functions of the two types of learning and, more importantly, to understand how the two learning threads cooperate and control the system learning.

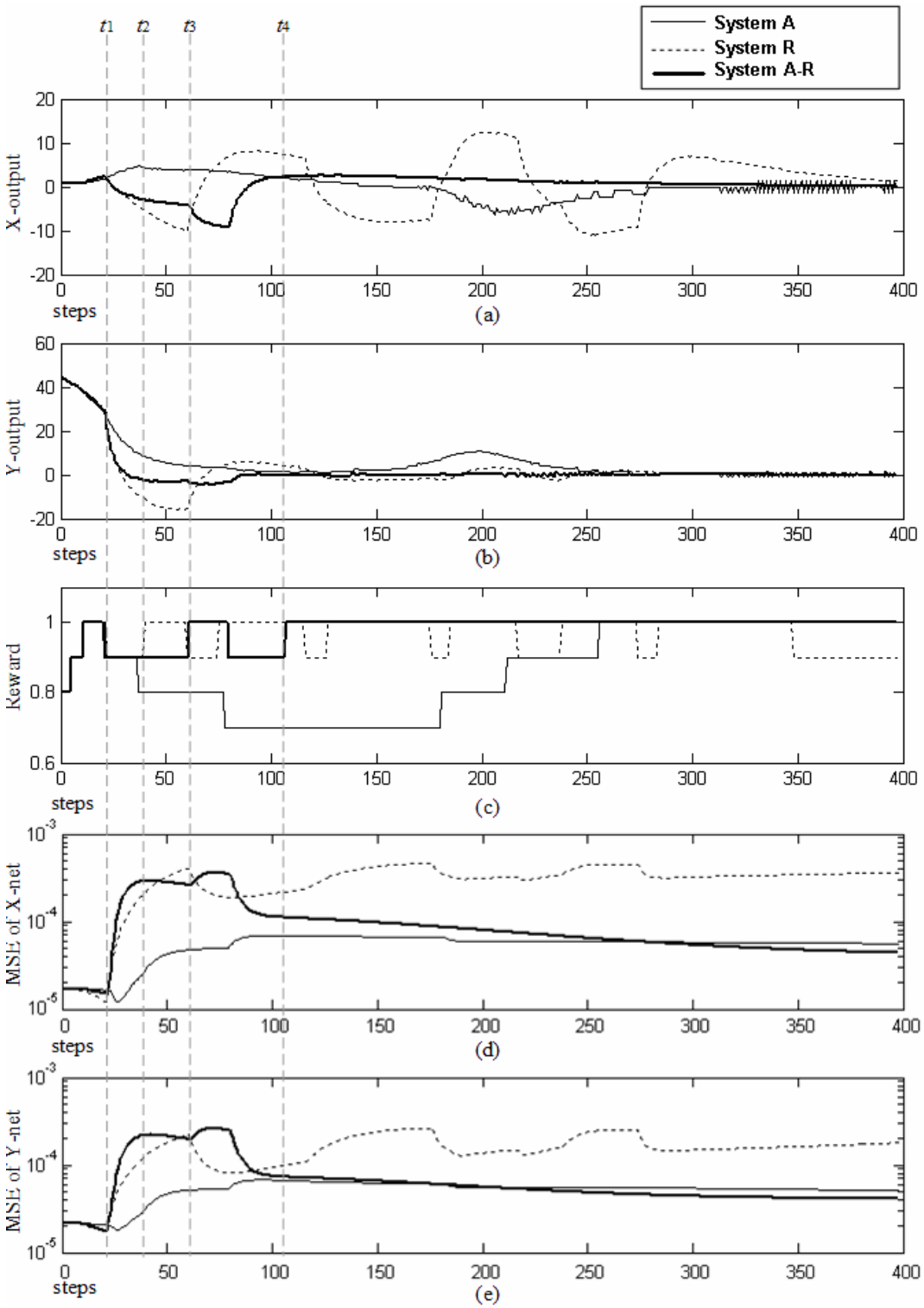


Figure 6-13 A comparison of the system measurements. Three systems: system with adaptive learning only, system with reactive learning only, and system using both learning threads are presented. The measurements describe the first walk facing the new peak where the old peak is recognised correctly by the system weights prior to the simulation.

Figure 6-13 gives the comparison of the network outputs and feedback history of the three simulations, on which our further discussion of system behaviours during the operation is based. It can be seen in Figure 6-13 that before time point t_1, all three

systems performed in a similar way, which was following the initial system output surface. At time t_1 the reward feedback of all three systems dropped. This drop happened when the robot went through the lower peak and moved out of it. New training data that advised against the previous knowledge of the network started to be collected at this point. The learning of the three systems then started to function actively and the differences between the three systems began to emerge.

System A

Despite the arriving of the new training data and the decrease in the reward, System A didn't have any obvious adjustment in the outputs. Its reward feedback kept dropping as the robot was moving away from the lower peak and heading towards the upper peak that is the peak previously recognised by the system weights. The reward feedback of this system started to increase around the 200^{th} step. This indicates that the robot was in the region of the upper peak. It is then obvious that the system is merely acting based on its previous knowledge and no efficient adjustment has been made because of the exposure of the new reward peak.

The adjustment of the system offered by adaptive System A was the smallest among the three but it had shown some efforts to slow down the robot around time t_2, though it didn't stop the robot moving away from the new peak. The outputs of System A change gradually and the MSE of System A was increasing slowly. These indicated that adaptive learning can only make the system adjustments to the new features slowly. This adjustment was not enough to steer the robot towards the new peak. Thus System A failed to achieve the goal in this walk and showed its inability to be using-while-learning in the simulation.

System R

After the new data were collected after t_1, System R had obvious adjustments of the outputs, trying to stop the robot from moving further in the wrong direction. These adjustments were based on the new training data but were against some of the existing data. As a result, the system MSE of System R increased significantly.

System R, using reactive learning only, showed a much quicker response to the change of the reward surface than System A. Actually, System R had the steepest adjustment of the system output and therefore was the first to alter its way back to the lower peak shortly after t_2. However, it showed no sign of slowing down because the

new incoming data supported the current system performance and the reactive learning offered little adjustment of the system. The system went out of the lower peak at t_3. Although System R re-entered the lower peak again soon, as it can be seen in the reward history, it had had little adjustment to slow down the robot while it was in the lower peak. It then went out of the peak soon after t_4. It can be seen from the system outputs that the reactive learning responded immediately trying to turn the robot back.

This huge overshoot of System R resulted in a large oscillation in the system output, as it is shown in Figure 6-13(a). From the reward history, it can be seen that the robot was going in and out of the lower peak many times. The MSE of System R increased significantly at the start of the walk when the robot stepped into the lower peak but MSE didn't decrease when the system progressed. On the contrary, the MSE kept increasing. The output oscillation and the increasing MSE then clearly suggest that System R, with reactive learning only, can react to the change of the environment fast but can't stabilise the system. System R learns fast but can't converge, which is hardly acceptable for the requirement of using-while-learning.

System A-R

After the new data were collected after t_1, System A-R had adjustments of its outputs similar to System R to stop the robot from moving in the wrong direction. A similar increase in MSE was also present. The magnitude of the system adjustment by System A-R was much stronger than System A but smaller than System R. After t_2, the reactive learning in System A-R was also strongly stimulated by the data it collected on learning the lower peak at t_3. It had the problem that the system didn't slow down the robot in time, similar to System R, and the system went out of the peak again. The speed of the robot changes more slowly than System R because the adjustment made by the reactive learning thread was compromised by the adaptive learning thread. System A-R re-entered the lower peak after t_3. The training data collected at this entry again stimulate the reactive learning where the robot started to move faster and the system MSE increased.

Soon after t_3, System A-R drove the robot out of the desired peak as well. The reactive learning thread was as effective as System R because the system changed the gradient of the output immediately, trying to turn the robot back. The training data collected at this point provided final stimulation to the reactive learning. After this

reactive adjustment, the system found the weights fitting both the existing training data and the new ones. This was the reason that the descent of the MSE became steep. The correct lower peak was then recognised by System A-R as the robot soon entered the correct peak at t_4 and started to stabilise itself. The entry to the peak this time had no impact on MSE because the new data matched the existing system function, meaning that the reactive learning was not strongly stimulated. Therefore the MSE didn't increase. After t_4, System A-R started to stabilise the robot at the peak and managed to do so at step 270.

6.3.5 Overview of Systems' Performance

System A in this simulation has shown clearly that the adaptive learning adjusts the system slowly. Recalling the simulation in Chapter 5 and Section 6.1.2, the only case where the adaptive learning can alter the system fast is the start of the learning where the size of training data set is very small and the system has not been well developed. The training of the adaptive learning is a compromise among all the existing data. This prevents its adjustment to favour any regional cluster of data. In this way, a good generalisation of the system can be achieved eventually. However, as a using-while-learning system, it also means it will be difficult for adaptive learning to act fast to the new features in the environment, which are started as small regional clusters of new training data.

Compared to System A, the network behaviour of System R is at the other end of the scale. Opposed to the adaptive learning, the reactive learning only favours a small regional cluster of data, the newest data in a given window length. The initiative of introducing this learning is to make the system react fast to the change of the environment, where the changes are represented by the new training data that trigger high system errors. In this section, the reactive learning has demonstrated its ability to alter the system function quickly. System R turned the robot into the lower peak shortly after it was discovered. However, the system failed to stabilise and started large oscillation. Because the adjustment of the reactive learning was so large, it constantly overshot. Large learning rates and constant steep adjustments of weights have a high risk of unsmooth system function and a low potential for good generalisation. It can be understood that such risk is even higher when the reactive learning overlooks the majority of the training data.

System A-R is a combination of System A and System R. In the system output history, it is clear that the change of the system made by System A-R is larger than adaptive learning and smaller than reactive learning. Because the system has both learning threads, they compromise with each other. When the data pattern presenting new features first comes in, the reactive leaning adjusted the system in a different direction from the adaptive learning. The system was mainly controlled by the reactive learning as it has a stronger adjustment of the system than adaptive learning. But the adaptive learning still worked towards the general best fit over all training data and discounted the dominant effects of reactive learning. In the simulation, the adaptive learning always regained the control of the system quickly when the robot moved to a flat area where no training data were available and reactive learning lost effect. In the error performance of System A-R, the steep change of error performance is always associated with dominant reactive learning. When the gradient of MSE is small, the adaptive learning has more effect in the system. This switch of dominance continued until step 70 when new data didn't cause the MSE to increase, meaning that the data were supported by the system function. By that time, the reactive learning stopped being strongly activated by new data and the adaptive learning took over the system entirely. The system then switched from the phase of adjusting to the phase of tuning.

6.4 The Complete Simulation Using Two Learning Threads

The simulation here is then a re-test of the learning scenario in Section 6.1.1. The introduction of the reactive learning thread is expected to improve the problem of learning revealed by Section 6.1 on a previous learning system so that the social positioning behaviour can adjust itself correctly and quickly when facing the new feature of the reward map to produce a feasible using-while-learning performance.

The positioning system with two learning threads was first operated with the map restricted to the upper half only for 40 walks. Then the robot was placed as the lower half of the map for 20 walks. Appendix III gives the full simulation history and measurements for this simulation. Figure 6-14 gives some examples of walks during the simulation. The reactive learning window length is 10 steps.

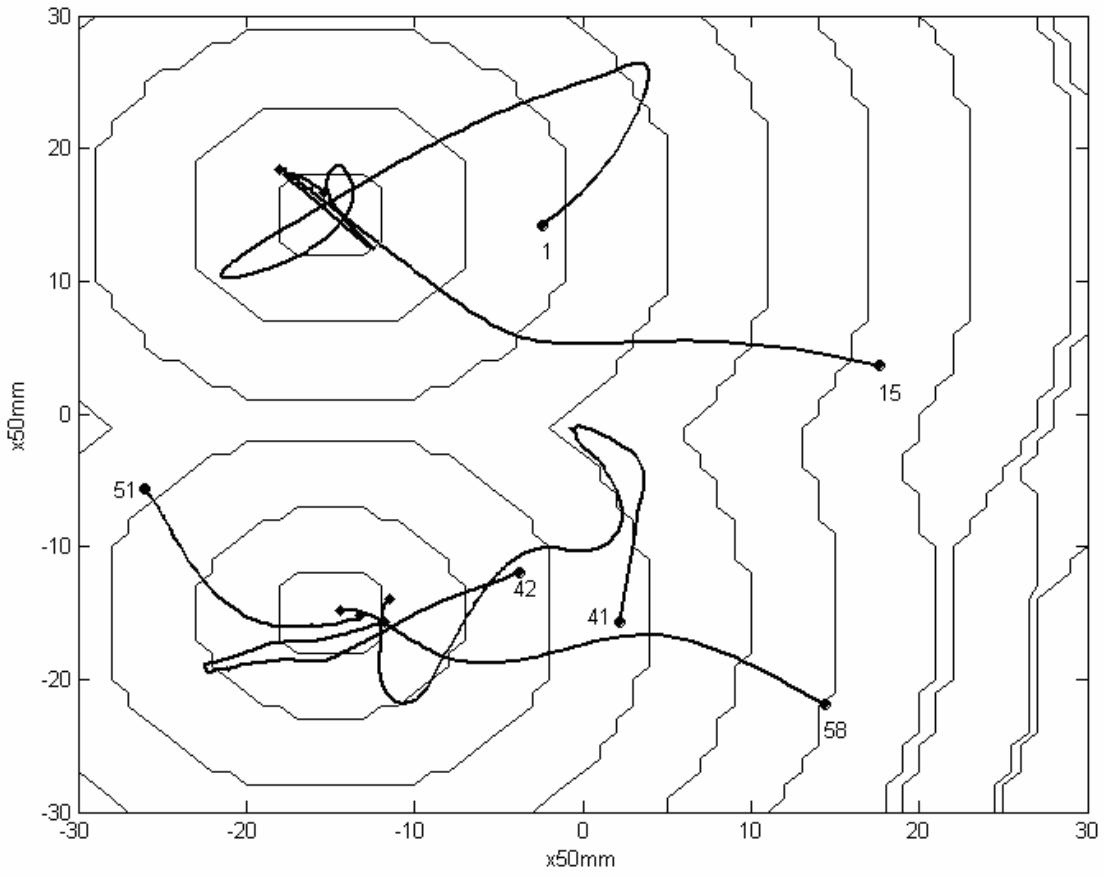


Figure 6-14 Some walks of the system simulation with both adaptive and reactive learning. The system faces the new lower peak after 40 walks of learning on the upper half of the map.

In Figure 6-14, it can be seen that the first walk of the system started near the position where the adaptive system in Section 6.1 started. But this walk here was successful because the reactive learning had stronger adjustment of the system. In the lower half of the map, where the last 20 walks are located, we can see that the system successfully changed the direction of the robot's movement at the beginning at the 41st walk with a clear reactive adjustment. Although this walk ended right beside the range of the lower peak, the system found the peak in the next walk. Later walks shown in Figure 6-14 are successful. Figure 6-15 shows the system measurements for this simulation.

From the system performance, it can be seen again that introducing the reactive learning with adaptive learning doesn't damage the system's generalisation and stability. The oscillation at the end of 40 walks is still only about 1% of the output range. Comparing the system response with Figure 6-5, it can be seen that with reactive learning, the system actually forms the correct output surface faster as can be seen in the reward feedback history.

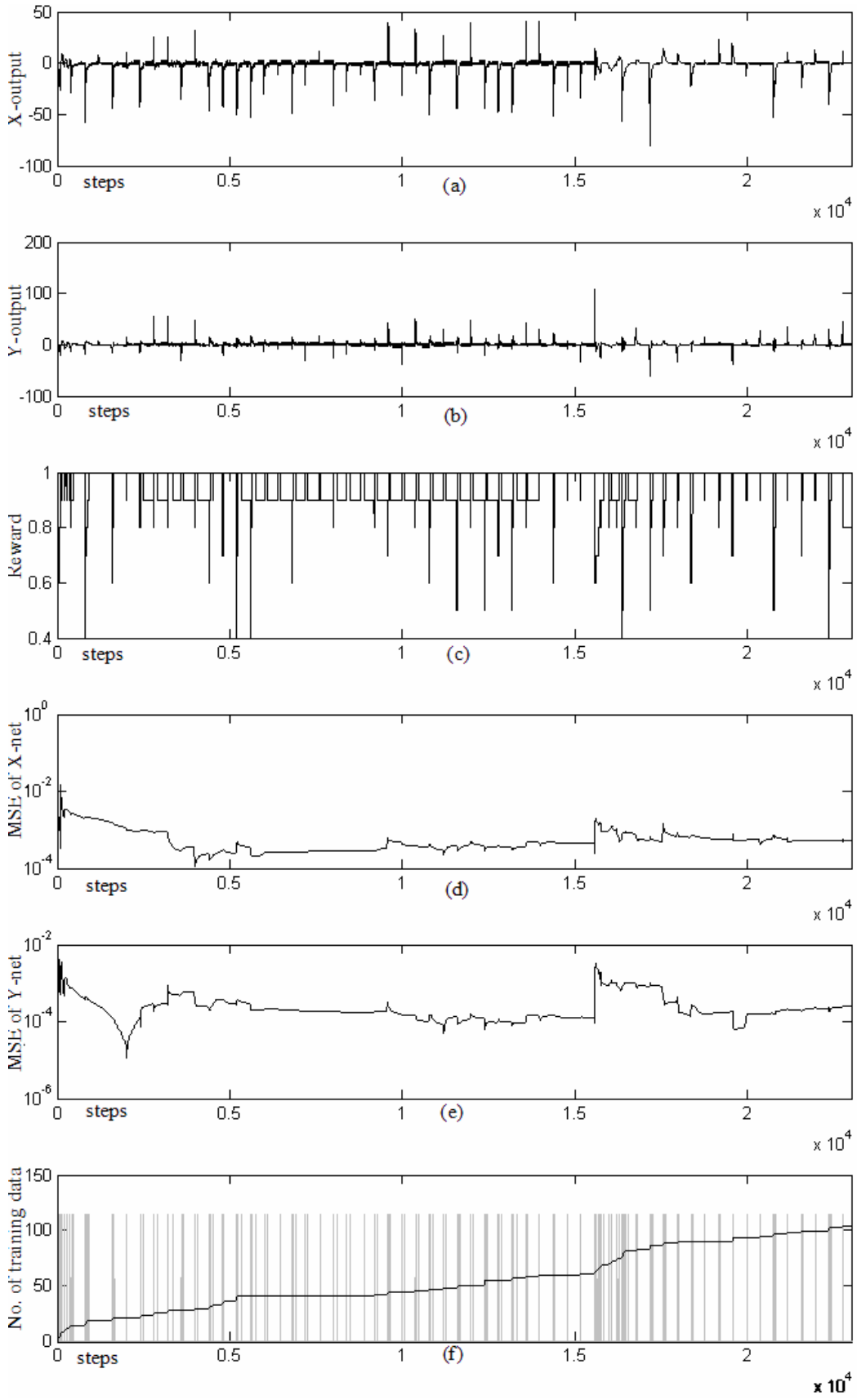


Figure 6-15 The system measurements of the simulation with system using both adaptive and reactive learning. The system faces the new lower peak after 40 walks of learning on the upper half of the map. The measurements show a total of 60 walks.

The reactive learning is triggered by the new incoming training data and the error measurements of those data. It then can be noticed that the steeper changes of the error measurements are associated with the updates of the training data. However, the learning will naturally switch the reactive learning down when the system has well developed weights so that even the most recent collected data are supported by the system output surface and thus give small errors. In that case, reactive learning has little adjustment of the system weights and the adaptive learning therefore will have larger effects on the adjustment of the learning system because it is engaged with more training data. Also, when the training data are not updating, reactive learning doesn't have any effect at all.

When the system finished 40 walks, the robot was then placed in lower half of the map. The system then runs in the lower half map for 20 walks. It is evident from Figure 6-15 that the system with both learning threads reacts to the change of the environment well and fast. Only two walks ended up with a local minimum near the goal and it recovered shortly in the next walk. From the output of the Y-network, it can be seen that the system doesn't suffer from the oscillation problem anymore. It can be noticed from the reward feedback history that the system has successfully recognised the direction of the new peak. Comparing the error performance of the system with the one in Section 6.1.1, it is clear that after the robot started in the lower half of the map, with the aid of reactive learning, the error drops down quickly, where the previous system's error performance barely falls at all. The difference in the speed of reacting to the new feature of the environment can also be illustrated by comparing the first walk in the lower half of the map of the systems with and without reactive learning. With reactive learning, the system adjusted the movements of the robot with a stronger force. The final output surface of the system in Figure 6-16 proves that introducing the reactive learning doesn't influence either the system's ability to generalise or the system's recognition of the existing reward peak.

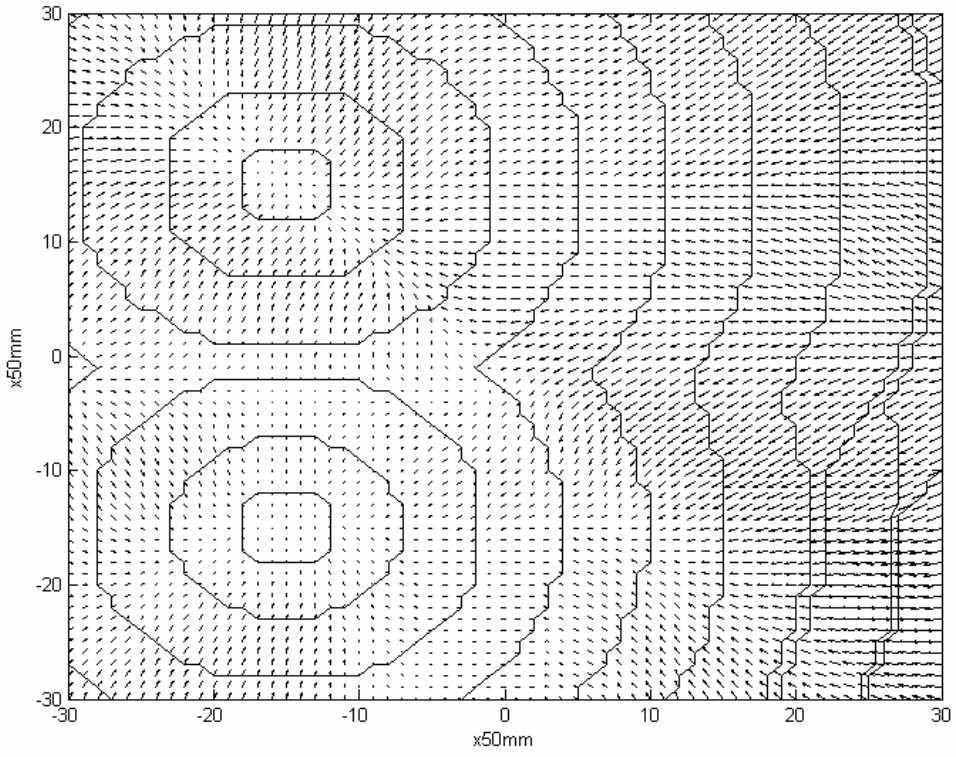


Figure 6-16 Output of the system using both reactive and adaptive learning, which faces two reward peaks in the reward surface.

Figure 6-17 is the integrated system output surface. Compared to Figure 6-4(b), the lower half of the integrated output surface has a clear and defined peak presenting the lower peak of the reward surface.

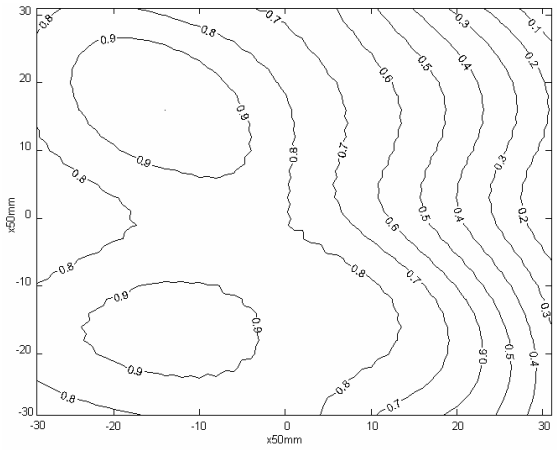


Figure 6-17 Integrated output surface of the system using both reactive and adaptive learning which faces two reward peaks in the reward surface.

The introduction of an extra reactive learning thread focusing on the local descent of the error surface has clearly given the system the benefit of reacting to the new changes of the environment quickly and contributes to the learning speed of the

system. It has then satisfied the requirement of the proposal of using-while-learning behaviour for the social positioning task.

6.5 Teaching the Robot: An Interactive Game

The simulation in Section 6.4 has demonstrated the system's using-while-learning ability. The fast learning and good generalisation have been seen when the system faces the problem of learning new features on the top of the previously learnt knowledge. However, the reward surface built for the simulation may not include all the complexities that a human might introduce.

While humans are involved, there are two major issues that previous simulations didn't cover. Firstly, when and where they choose to provide trainable information, e.g. a change of the reward value, is unknown. This means the flat area in the reward surface becomes dynamic. Secondly, how many levels of feedback the human will use effectively is unknown. The use of different levels of feedback, i.e. reward steps, is essential for the training data selection to filter the best movement of the robot so far for training. Fewer reward steps will affect the quality of the training data selected and then influence the learning of the system.

In order to investigate the ability of the proposed learning system under those problems, we introduce this interactive simulation where the human interacts with a simulated robot. There are only two levels of reward values available for the human to use, which is the minimum number of reward steps possible to rank the performance of the robot. The interacting system is designed to allow users to input their feedback using the mouse. Because the MATLAB GUI is not able to give the users a visual impression of human following, we introduced the simulator to the users as an interactive game in which the users were asked to teach the robot to get to certain position through a certain path in the human coordinate frame. A GUI interface, as shown in Figure 6-18, has been designed so that the human can interact with the simulated robot.

While interacting with the robot, the user can stop the simulation at any time when they feel satisfied by pressing any key on the keyboard. The trace of the robot movements is not visible to the user so that they judge based on their feelings without any mathematical aid. People were asked to teach the robot to move and stay at a certain point in the map through a certain path by clicking the mouse.

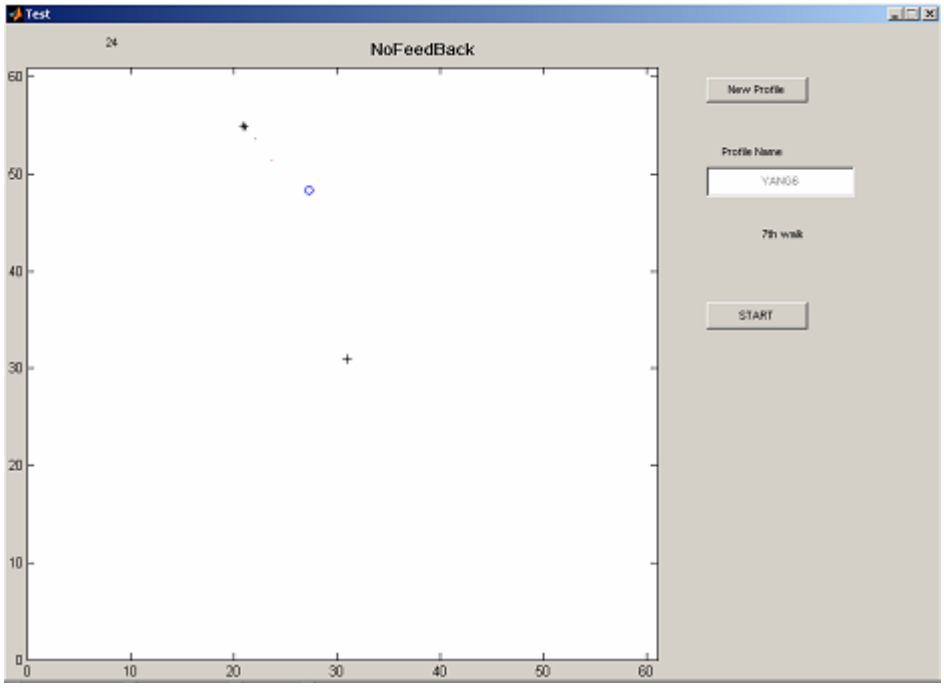


Figure 6-18 The GUI interface for the interacting game with the virtual robot. + shows the centre of the map, * marks the start points of the robot and ○ is the moving robot.

6.5.1 Further Modification of the Learning System

In this interface, the user can penalise the robot by clicking the left key or reward it by clicking the right key. This means that feedback input of the system is ΔR^k with only three possible values: -1, 0 and 1. When to give feedback depends on the user. Our investigations show that users are more willing to comment on the movement of the robot rather than its position.

In the previous training data selection method, the value of ΔR^k marks the degree of the gradient in the reward surface. The validation of better training data is based on a greater value of $|\Delta R^k|$. However, here, ΔR^k only has three values. The training data at time k thus is modified as:

if $\Delta R^k = 1$:

$$\mu_t(\hat{\mathbf{p}}^k) = \tilde{\mu}(\mathbf{p}^k) \quad , \tag{6-3}$$

if $\Delta R^k = -1$:

$$\mu_t(\hat{\mathbf{p}}^{k+1}) = -1 \times \tilde{\mu}(\mathbf{p}^k) \quad . \tag{6-4}$$

The use of $\Delta r_t(.)$ is not needed because there are too few feedback levels.

6.5.2 Results of Interaction Tests

Three people participated in the test. We found it common that users initially had difficulties interacting with the robot. Their feedback tended to be much noisier than the previous training data from a fixed reward surface. The main difficulty the users first encountered was that they panicked when the robot seemed to stop at a wrong location, which means that the system had converged into a local minimum. Their feedback then became chaotic and no effective learning could be established. However, after a few interactions with the robot, the users started to get used to the interaction and become patient. They realised that after some feedback data have been given, the robot may need a moment to adjust its movement.

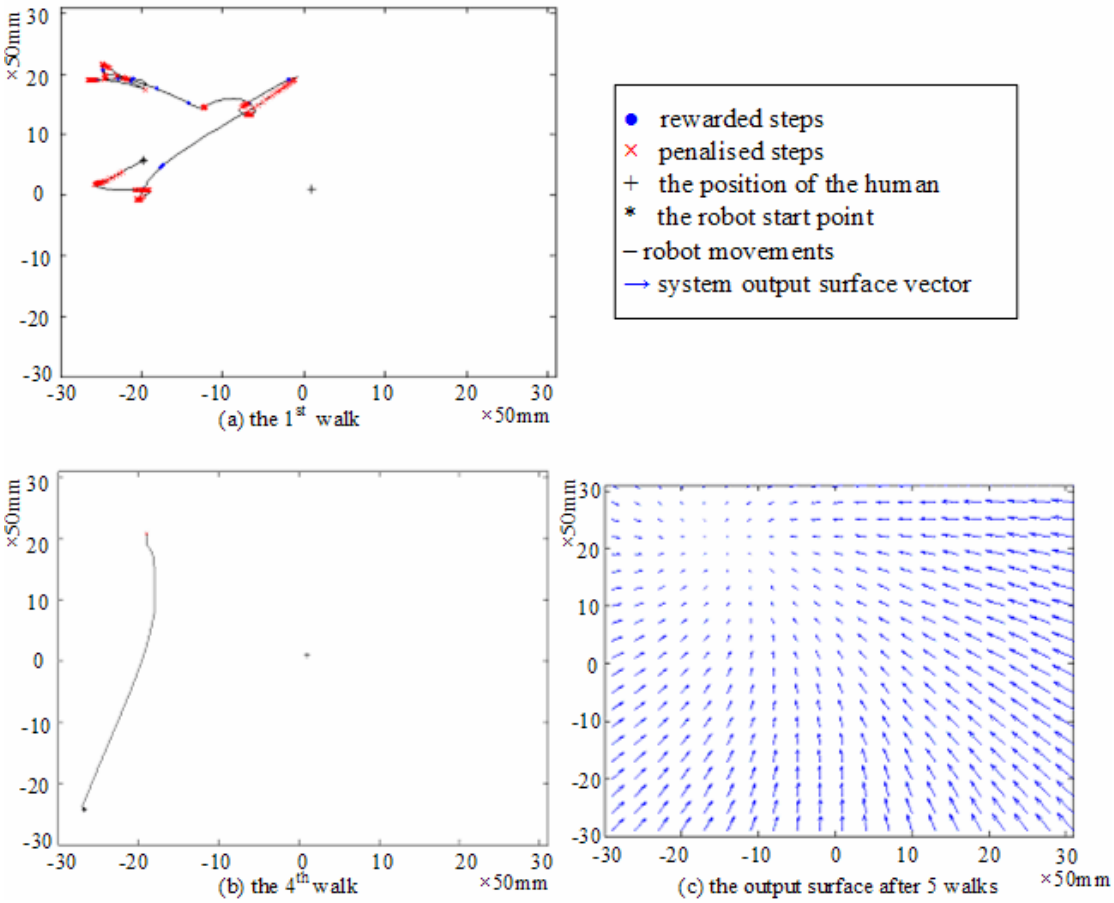


Figure 6-19 The test results of a user who has been familiar with the interaction and tried to teach the robot to follow at the left behind.

Figure 6-19 is the result of one test after a user became familiar with the interaction. He was trying to teach the robot to get to approximately [-1000, 1000] mm, i.e. [-20.20] in the map. Figure 6-19(a) is the first walk of the robot and Figure 6-19(b) shows the results after 2 more walks, where the user felt no feedback was necessary. Rather than reward it frequently, users appeared to reward it when it was making progress or improvement, partly because they found it is easier to teach the robot this way. When they thought the robot has fully learnt what they want, they would stop giving feedback. Figure 6-19(c) shows the system performance after the robot has learnt fully after 5 walks.

In the next test, we investigated the system's performance in the case that the human's preference changed while the robot had previously learnt knowledge. In Figure 6-19(c), we can see that the robot learnt to take the shortest routes in the map to reach the goal. The user then was asked to change the preference so that robot didn't head to the goal directly but went around through certain path. This is similar to the human-following case in which certain people don't like the follower to go across their front or back. Figure 6-20 shows the interaction results. This interaction is based on the system shown in Figure 6-19(c). The system was initialised with the weight basis and the training data set from the end of previous simulations demonstrated in Figure 6-19. In Figure 6-20(a), we can see that the robot managed to learn from the human though a small difference between the human's expectations exists. However, the system result in Figure 6-20(b) shows a reasonable adaptation.

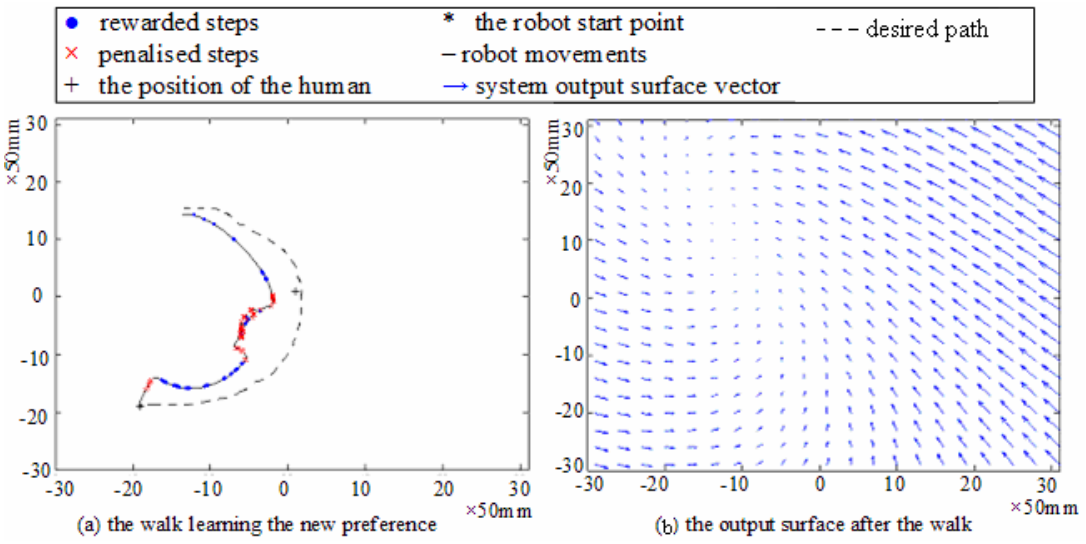


Figure 6-20 The test in which the user changes the preference in the interactive experiments.

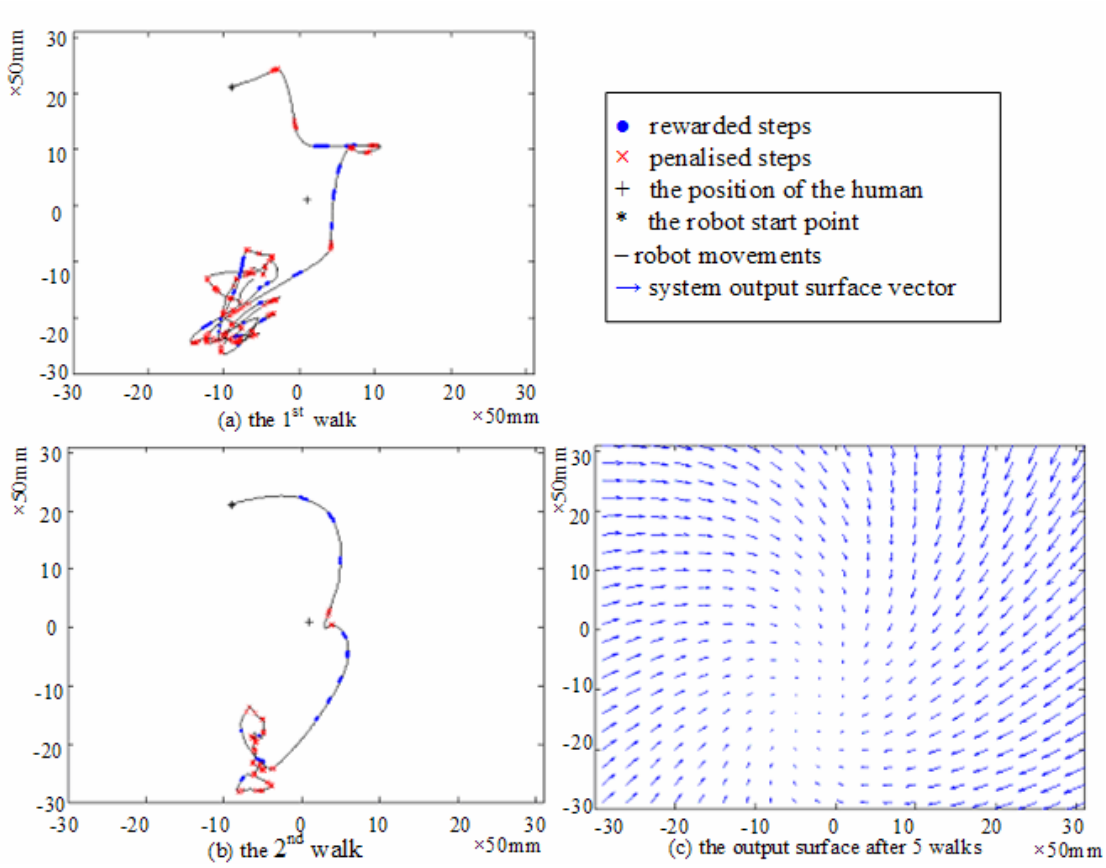


Figure 6-21 The test of a difficult preference with no previous weight basis, where the gradient of the reward surface is not a direct line towards the goal.

Figure 6-20(b) expresses a mapping that is more complex than Figure 6-19(c). Thus, is the system able to learn effectively in such a situation without the basis of Figure 6-19(c)? In third experiments, the user was trying to teach the robot to rich a point in the lower half of the map by going around the central point, where the system started with initial random weights. Figure 6-21 shows the results of the learning. Figure 6-21(a) demonstrates the first walk and Figure 6-21(b) is the second one. Figure 6-21(c) is the system performance after the user stated that he was happy with the robot's movement. The system shows the ability to learn effectively.

The interactions have been successful when the users got used to the way of interacting with the robot and a reasonable learning of the system has then been established. However, in Figure 6-19(a) as well as Figure 6-21(a) and (b), we can see that the robot needed a long time to learn the action to stop at the goal location. But the robot still manages to learn it in a reasonable amount of time (normally around 2000 steps in the first walk) if the human keeps sending feedback. Once it is learnt, the system performs well.

After the tests, the statistics show that the average feedback rate of users is about 8.4%, which means the human inputs feedback on 8.4 out of 100 steps. 2.4 steps of these 8.4 steps of feedback are rewarding and 6.0 steps are penalising. The human inputs can be described as sparse, noisy and uncertain. But the system shows the ability to adapt in such a situation and the interactions were then successful when the user felt happy with the robot's movement by not giving feedback anymore.

In this simulator, the human only has two usable feedback levels: 'good' or 'bad'. This is a very difficult situation for the system, as it provides no options to filter or validate better training data. This has greatly reduced the efficiency of the training data selection and the quality of the selected data is very low. When more levels of feedback are involved, which will be the case in the real robot experiments, the use of $\Delta r_t(.)$, will be then available again. Thus a better performance of the system can be expected.

6.6 Capabilities of the Learning System

In this chapter, a proposal of the modification of the learning system studied in Chapter 5 has been introduced. The initiative of this proposal is to secure the system's ability of using-while-learning when the environment and the human preference become more complex. The challenge recognised in this chapter is for the system to learn new features while retaining learnt knowledge and to maintain a low tolerance of failure, as required by using-while-learning systems. The study discussed two different learning methods: the adaptive learning that generalises well but converges slowly and the reactive learning that converges fast but generalises poorly. By combining these two learning threads into one learning system, the research successfully achieved a system that converges very fast while maintaining a good generalisation ability so that the system can learn new features at any point in the operation. Our argument has then been supported by our simulations. The interactive simulations we presented demonstrated the feasibility of adopting the system in real interactions with human.

From Chapter 3 to Chapter 6, we have studied the model and the learning system for social positioning behaviour. A complete using-while-learning system with a novel and effective framework has been proposed. Extensive simulations in a wide range of situations have been presented, the results of which supported the using-

while-learning ability of the proposed system. In the next chapter, therefore, we will discuss some of the key issues in the embodiment of the robot, the hardware, and the control platform we built for the social positioning behaviour to be completely implemented in the future.

Chapter 7 The Mobile Robot Control System

In Chapters 3 to 6, we developed the social positioning behaviour and proposed an autonomous online learning system for it. The behaviour has then been tested in simulations, and robust results were presented. In this chapter, we study the mobile robot control system that can implement the social positioning behaviour on the robot. The control system not only focuses on implementing the existing behaviour but also aims to build a platform into which further behaviours can be incorporated. This offers freedom to extend the system to includemultiple behaviours. The basic structure of the control system is shown in Figure 7-1.

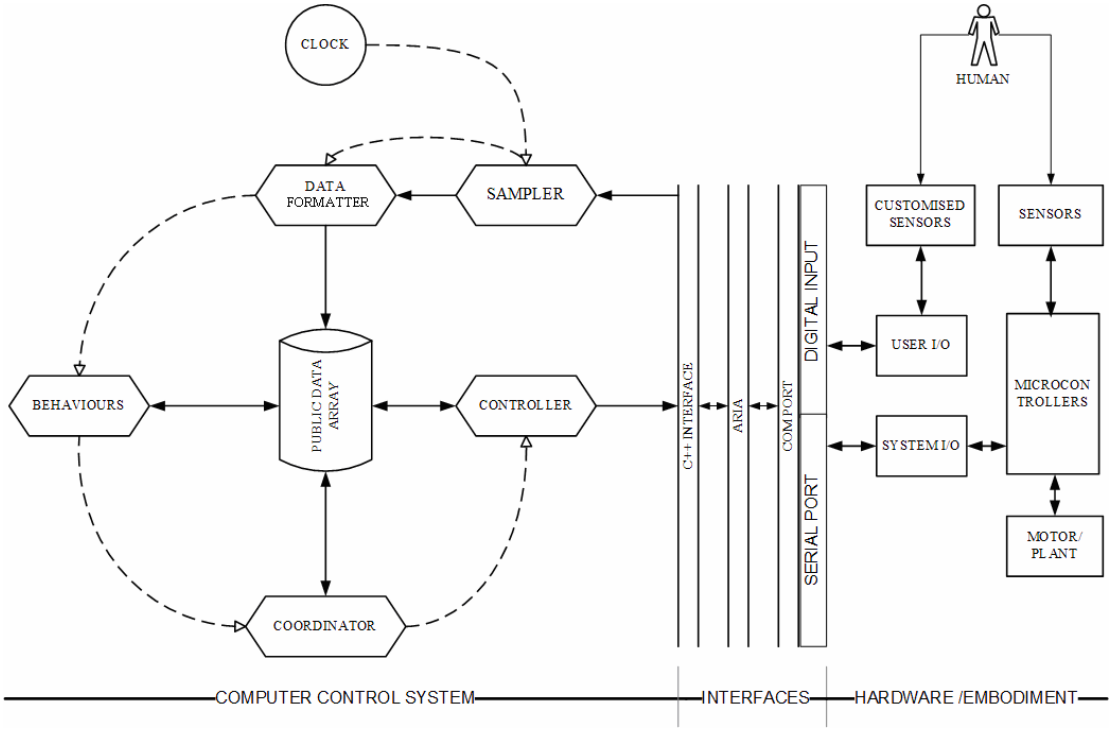


Figure 7-1 The mobile robot control structure of the interactive robot system. The solid arrows mark the data flow and the dashed arrows mark the system execution sequence in the computer control system.

The robot control system consists of three major components: the robot hardware, the computer interfaces and the control software. Section 7.1 introduces the hardware system and the embodiment of the robot. The human position sensor will also be discussed. Section 7.2 introduces the interfacing of the system between its software control system and the hardware. Section 7.3 introduces the structure of the software control system. Section 7.4 analyses the kinematics of the system. Finally,

Section 7.5 demonstrates the system's ability to control the robot by describing a sequence of experiments using a pre-learnt social positioning behaviour.

7.1 The Pioneer 2 Robot and the Human Position Detector

The robot used in this research is a Pioneer 2DX mobile robot. It is equipped with two DC motor wheels and a ring of sixteen ultrasonic rangefinders. The robot's physical dimensions and its top console deck are shown in Figure 7-2.

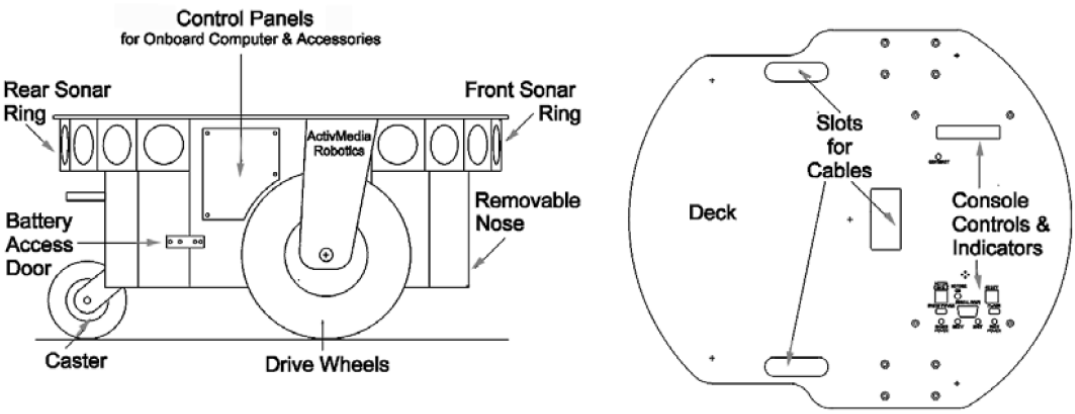


Figure 7-2 The physical dimension and top console deck of Pioneer II mobile robot (source: Manual of Pioneer II and Peoplebot, v11)

The Pioneer 2 operating system used in our research is version 1.P. The robot has a group of built-in controllers for motor and sensor control. The motor controller implements a control algorithm for the robot wheels and it has been suggested by the manufacturer not to bypass this controller. Appendix IV gives more details of the onboard controllers.

The social positioning behaviour is based on measuring the relative pose of the robot and the human. Previous research has proposed many possible methods to measure the position of a human using range sensors such as sonar and lasers, most of which are based on statistical and probabilistic methods (e.g. Thrun et al., 2001, Bueno and Kragic, 2006, Arulampalam et al., 2002). The ultrasound sonar detectors on the Pioneer 2, however, are not well-suited to this purpose. Besides the noisy input, it is difficult to use range detectors to distinguish which object is the human who is currently involved in the HRI, as the environment can have many moving objects in the background. In other words, the robot has no means to identify the human leader using range sensors. Sekmen et al. (2002) have proposed using Passive Infrared (PIR)

detectors for human detection. The overlapping sensing ranges of multiple PIRs divide the space into several regions. People in different regions will trigger a different set of PIRs. This technique, by recording the combination of triggered PIRs, can not only detect moving people but also estimate their distance. In our efforts to reproduce a similar system, a set of PIRs, KC7783R, were used. Each KC7783R covers a region approximately 60^o wide with a 2 m radius. However, after the PIRs were mounted on the robot, they gave an unacceptably noisy response. The KC7783R is built to be most sensitive at the peak length of the human radiation spectrum. Unfortunately, we found that many objects have a similar radiation spectrum peak length. The problem becomes more serious when the radiators in the environment are turned on. Therefore, these PIRs were felt to be unlikely to be a reliable method for detecting human movement.

Thus, the current research considers detectors that can be attached to the target human, and which can give a direct measurement of the position of the human relative to the robot. The research uses a GameTrak™ motion sensor to complete the position measurement. The GameTrak™ sensor was originally marketed as a game console controller. It is designed to detect the hand movements of the player so as to control the game in an interactive manner. The sensor is shown in Figure 7-3. It consists of two identical detectors, each of which has a flexible wire that can be attached to the human. The wire is connected to the sensor through a rotational base. The outputs of the detector will give readings to identify the exact position of the point attached to human. Only one detector has been used for human position measurement. The outputs of the sensor range from 0V to 5V, where the outputs are all zero for maximum measurements and all 5V for minimum measurements. By using ten-bit quantisation in the robot I/O, the device offers a resolution of approximately 2 mm. Appendix V gives the details of the model of the device.

This device can provide the position of the human relative to the robot, denoted by the vector **q**. It should be noticed that **q** is the human position vector in the robot coordinate frame. However, the required input data for the social positioning model is the robot position vector, **p**, in the human system (see Section 3.2). The transformation between the two vectors is discussed in Section 7.4.

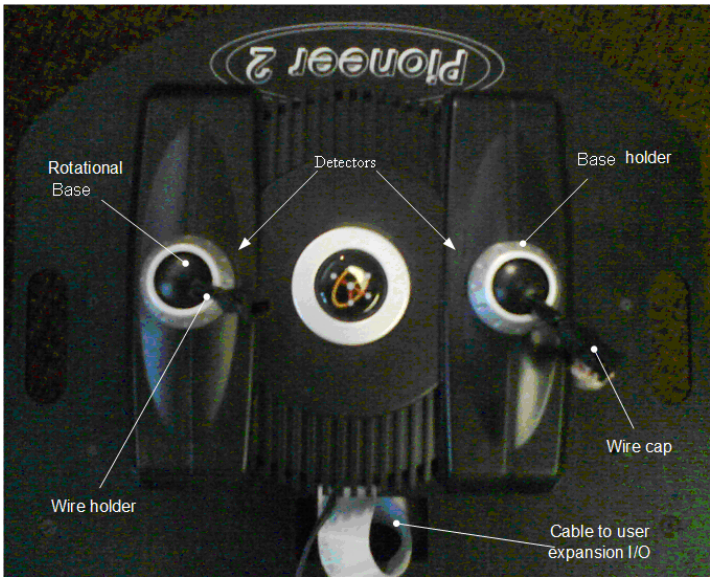


Figure 7-3 Top view of GameTrak^{TM} motion senor with comments on the components.

7.2 The Computer Interface

The communication between the computer and the robot's on board control system can be established through the COM port by calling ARIA functions. ARIA is the class of functions introduced by the manufacturer for interfacing to Pioneer robots. ARIA is compatible with most major computer languages but the manual recommends VC++ 6.0 for highest efficiency.

However, ARIA objects can only be called from console applications in VC++ 6.0. In order to provide more operability and user-friendliness to the control program, only a simple console core application was created to enable effective communication between the computer and the robot. A Windows GUI based control application, which contains the computer control system, was then built on the top of the VC++ interfacing program in VB 7. VB was chosen to give a shorter development cycle.

7.3 The Computer Control System

As shown in Figure 7-1, the computer control system consists of seven elements: clock, sampler, data formatter, behaviours, coordinator, controller and public data array. The clock controls the system working frequency. During each working cycle, the sampler, the data formatter, the behaviours, the coordinator and the controller all

execute in sequence as shown by the dashed arrows in Figure 7-1. Each function block doesn't communicate directly with other functions but stores data into a public data array, to which each function has access. The only exception is the communication between the sampler and the data formatter. The functions are not run in parallel in order to minimise the system overheads so that a higher working frequency can be achieved from the computer system. As each function visits the data in a certain order, the data access does not need to be scheduled.

In order to decide the system working frequency, the first concern is the control of the motors of the robot. A high enough frequency has to be used for robust control. The research uses 10 Hz. However, another concern is that 10 Hz is a high frequency relative to the speed of the movements of a human and a robot in a normal office environment. The observable change of their postures within 0.1 second is small and can be easily covered by noise. Therefore, the system computation selects a time window with a certain length. At any time instance, the robot and human actions are computed based on movements within this time window, as detailed in Section 7.4. At any time instance, behaviours that require sensor readings at the previous sample point in the computation model use the data that date back the whole time window length instead. In this manner, robustness is ensured in both the control and the computation.

When the system clock triggers a thread, the sampler is the first function to be executed. Its responsibility is to read the data from the robot system, including the motor speed and sensor data, by calling the VC++ console application. The sampler will then pass the data directly to the data formatter, where the raw sensor data will be modelled and necessary data transformations will be calculated. The processed data will then be stored into the public data array and the sampler will enable the behaviours. The behaviours are initialised by visiting the public data array to access their input data and they subsequently store their results into the public data array. The next function in the process line is the coordinator. It will combine the outputs of all behaviours into one desired target for robot movement. The behaviours that will be implemented in the experiments in Section 7.5 are social positioning and emergency stop, and the coordinator's task here is therefore to decide whether the outputs of emergency stop need to override the outputs of social positioning. After the desired robot movements have been stored into the public data array, the controller is the last function to run. It calculates the control signal for the robot, based on the desired

target position and then calls the VC++ console application to send the command to the robot system.

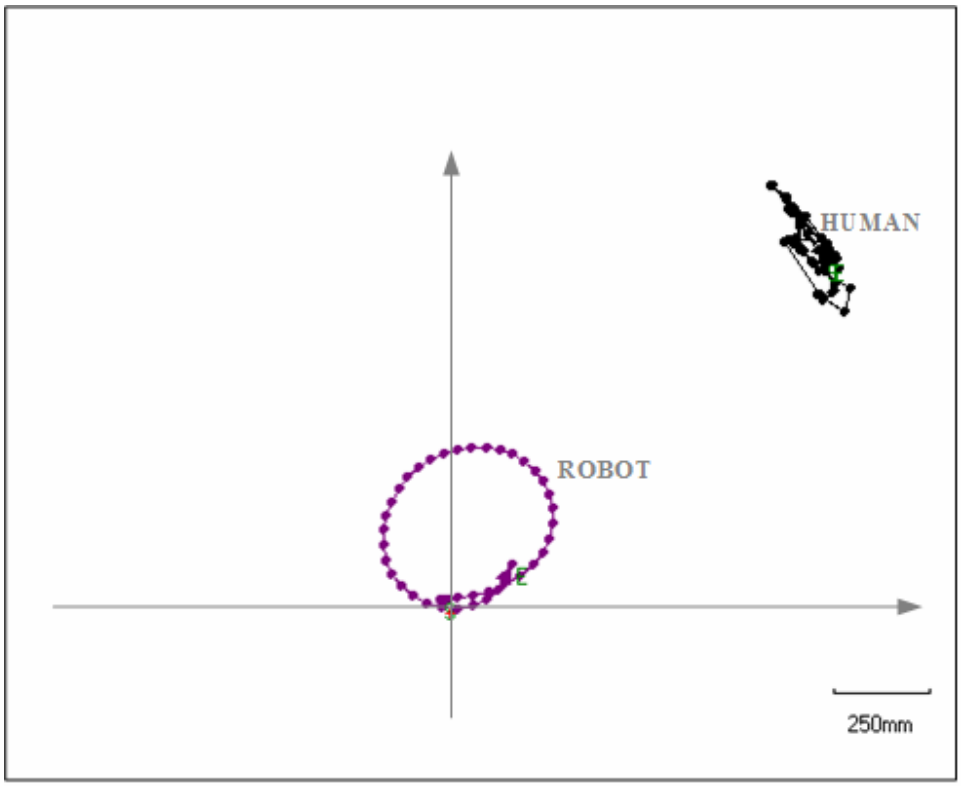
In the computations of the system, however, it has to be noticed that the measurement of the human position is based on the local robot coordinate frame, which rotates. Therefore human position vectors read at different time instances can't be used in the same computation directly because all vectors are observed from different coordinate frames. Global coordinates can't be measured directly as there is no global positioning sensor. Therefore, the kinematics of the system during operation have to be analysed for the purposes of global system tracking and coordinate normalisation as well as for computing human movements, the details of which is located in Appendix VI.

7.4 Experiments

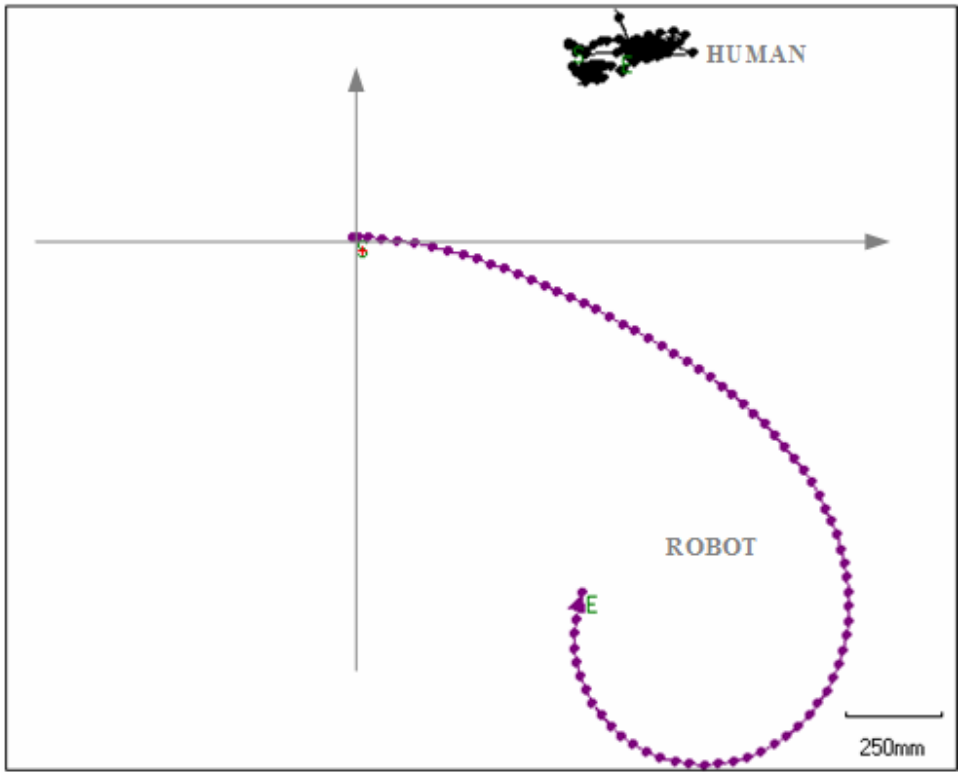
In order to test the accuracy of the system measurements, experiments were carried out with a stationary human.

In these experiments, the robot was assigned a set of pre-defined movements and the aim of the experiments was to test if the system could correctly interpret the raw data from the sensor as well as tracking the relative movements of the human and the robot. The time window length in the experiments was 5, the value of which has been empirically justified. Figure 7-4 displays the results of two of these experiments, in which the measurements of the human positions are expected to be stationary. The human measurements however appears to be a small cluster of points. The range of the cluster is about 300 mm, which gives a measurement error of $\pm$ 150 mm. This is less than half of the width of the robot and is considered to be acceptable.

In the next experiment, a pre-learnt social positioning behaviour was installed into the system. The controlling methods were as discussed in Appendix VI. The results are shown in Figure 7-5, in which the robot doesn't learn and faces right initially, and the human stays still and faces right. The robot follows the learnt positioning surface correctly in Figure 7-5. The true movement of the robot is illustrated in the global view of the system, Figure 7-5(a). The jerkiness of the movement in Figure 7-5(b) is introduced by the measurement errors of the human position as the robot movements in the human reference frame are actually the difference between the robot and human positions in the global frame (Figure 7-5(a)).



(a)



(b)

Figure 7-4 System measurements tracking with pre-define robot movements and stationary a human. The robot starts at the origin of the coordinates.

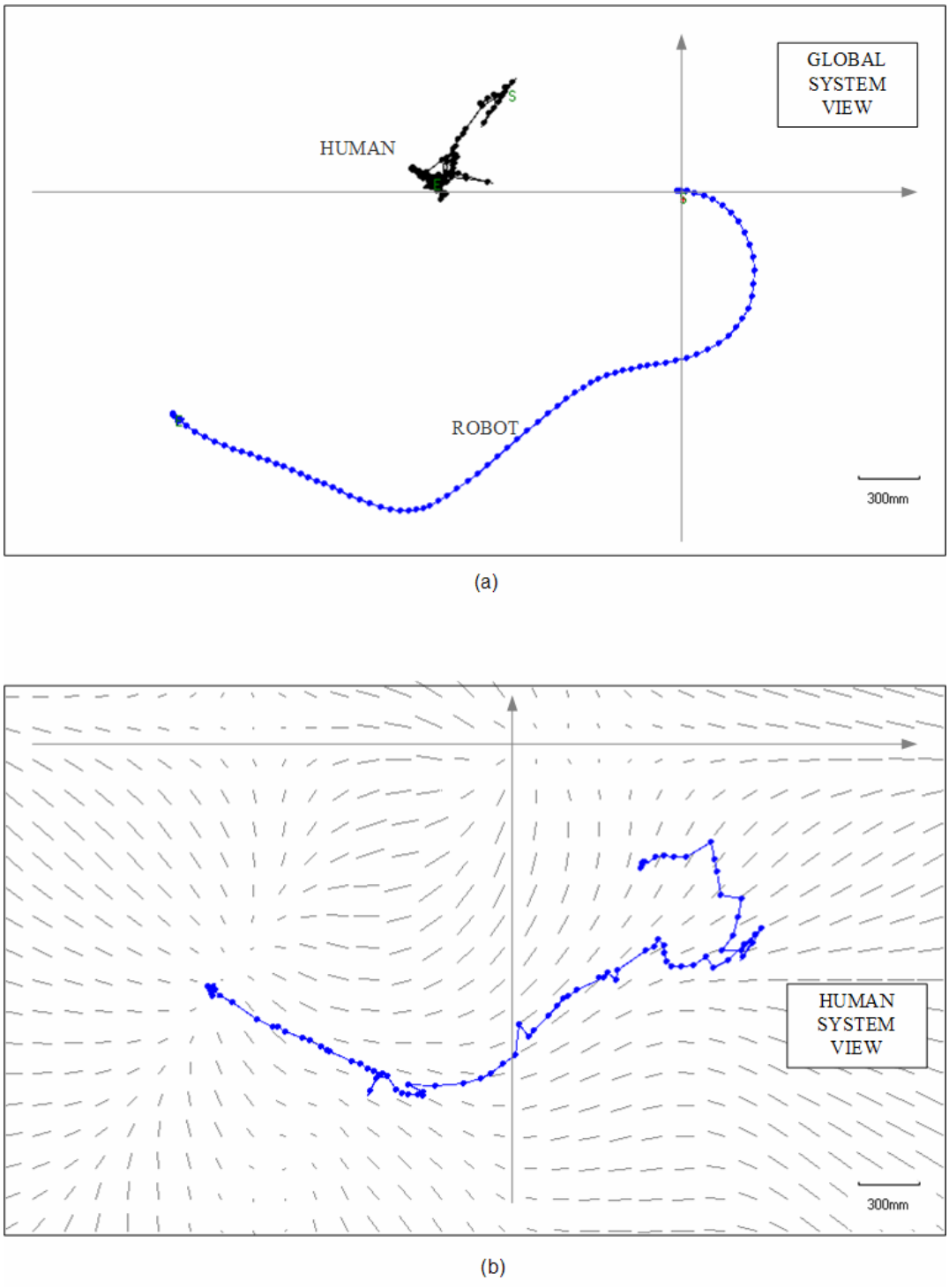
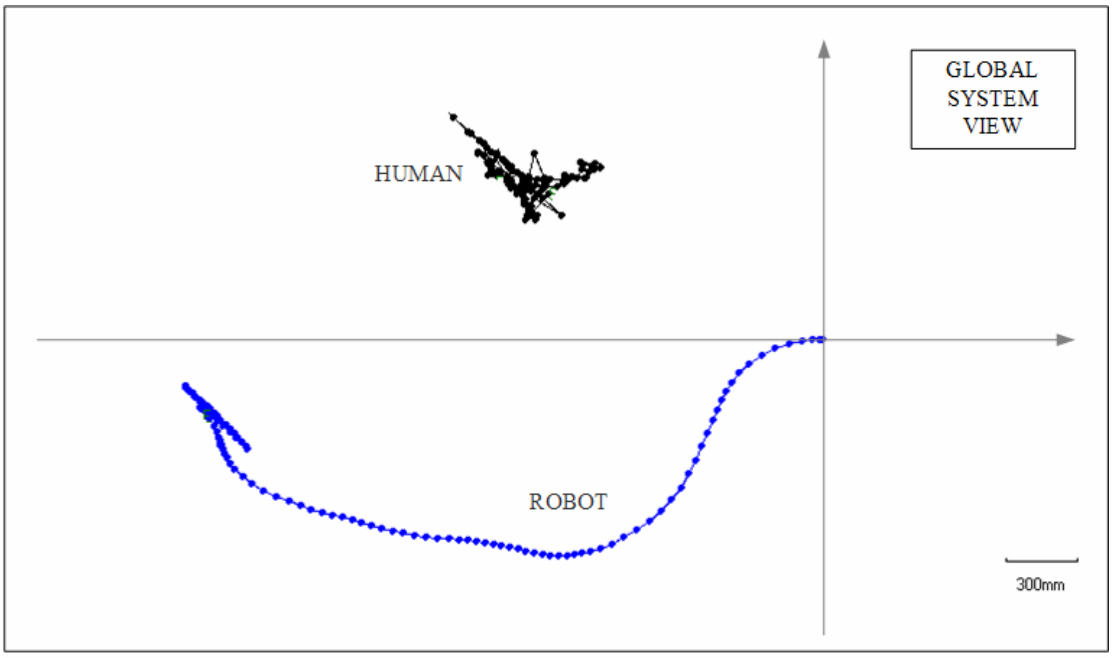
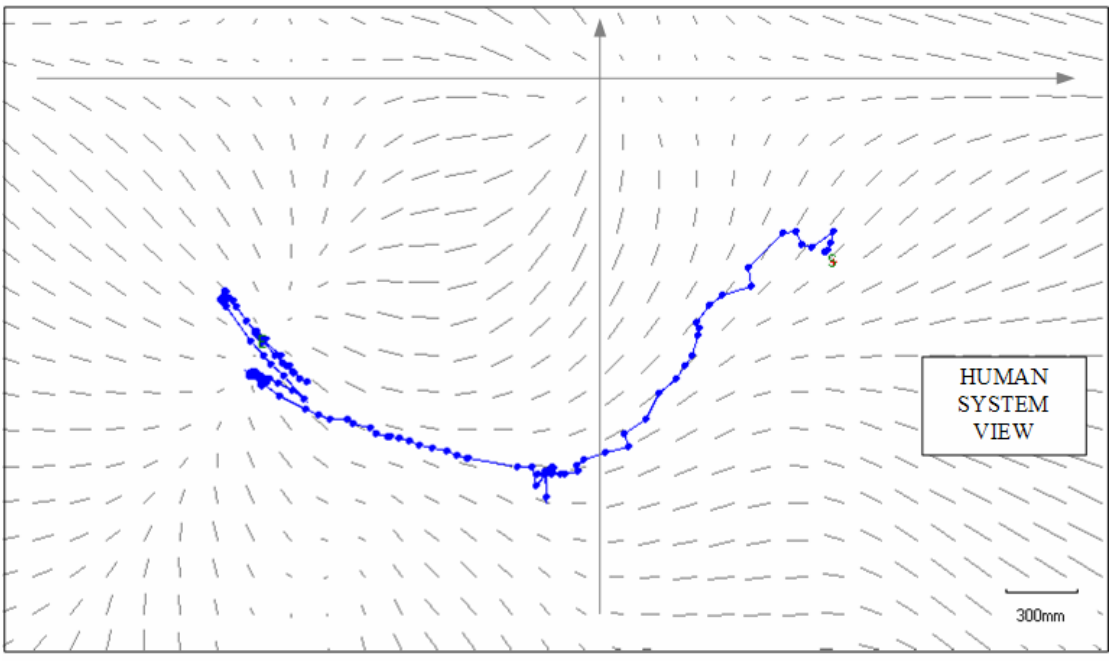


Figure 7-5 System experiments with pre-learnt social positioning behaviour and a stationary human where the robot is not allowed to move backwards. (a): the true movement of the robot. (b) the movement relative to human. The robot starts at the origin of the coordinates.



(a)



(b)

Figure 7-6 System experiments with pre-learnt social positioning behaviour and a stationary human, where the robot is allowed to move backwards. The robot is moving backwards throughout. (a): the true movement of the robot. (b) the movement relative to human. The robot starts at the origin of the coordinates.

It can be see that the robot has to go through a long distance to turn around. This distance is long because the control signal has been scaled down to ensure robust control of the rotation. A simple and reasonable way to improve this situation without modifying the control algorithm is to allow the robot to move backwards. Let $\theta = f\angle\mathbf{s}_d^k$. The control signals are updated differently from Equation (VI -11) in Appendix VI when $|\theta| > \frac{\pi}{2}$:

$$v_d^{k+1} = -c_v f\left\|\mathbf{s}_d^k\right\| \qquad (7\text{-}1)$$

$$\omega_d^{k+1} = \begin{cases} c_\omega(\theta - \pi) & \theta > 0 \\ c_\omega(\theta + \pi) & \theta < 0 \end{cases},$$

where c_ω is 0.5. When $|\theta| \le \frac{\pi}{2}$, the formula is the same as Equation (VI -11). The experiment that allows the robot to move backwards is shown in Figure 7-6. The robot is moving backwards throughout the experiments shown in Figure 7-6. It can be seen that the movements of the robot remain correct without the large turning around at the beginning, which improves the efficiency of the adjusting of robot's position and achieves the goal in shorter time.

The results of the experiments have clearly shown the system's ability to control the robot by using a learnt social positioning behaviour. After the accomplishment of this task, the robot platform is ready for further development to introduce the human feedback device and other behaviours to enable the robot to finish a comprehensive human-following task. The core of the human-following task is the social positioning behaviour that has been intensively studied in this thesis. The using-while-learning behaviour combined with other research results on human feedback devices (e.g. Triesch et al., 2006, Pacchierotti et al., 2006, Mitsunaga et al., 2005, Nakauchi, 2002, Roy and Pentland, 2002) and other robot behaviours (e.g. Hoffmann, 2004, Song-Yee et al., 2000, Breazeal, 2000) from the literature make further development towards a more comprehensive human-following task possible. However, what has been revealed by the study of this social positioning behaviour goes beyond the proposed HRI scenario. In the next chapter, we will discuss the generalisation of our study after a summary of the contributions of this thesis.

Chapter 8 Conclusion and Future Work

This thesis is devoted to solving the problem of an autonomous learning of social positioning behaviour that is the core interactive behaviour to accomplish the human following task. After the review of the context of the research in Chapter 2, the main challenge of this thesis has been identified as the design of a using-while-learning system that can be trained using human feedback to position the robot correctly according to the human's preferred social distance.

The social positioning behaviour has been modelled as a transition among system states that are the position of the robot relative to the human. The preference of the human is represented by a surface among the system states: the reward surface. The objectives of the learning system are to determine the desired transitions among states and to locate the goal state based on a reward surface that is unknown until the system operates, as concluded in Chapter 3. After a review of the two possible ways of learning in Chapter 2, Chapter 4 describes the research into forming the social positioning behaviour with reinforcement learning. By introducing the secondary scoring surface and neighbour learning into reinforcement learning, the system learns successfully. But reinforcement learning can't avoid lengthy random firing and it can't learn states that the system has not experienced, which does not satisfy the high requirement of using-while-learning in a social environment. Chapter 5 therefore investigates an alternative: the multilayered feedforward network with error backpropagation. A novel online training data selection method has been proposed to enable the learning from human feedback. With a dynamic learning rate based on the error performance, the system produces promising results in simulations with a simple reward surface. Chapter 6 therefore places the system in a more complex scenario, where more features of the human preference can be explored during the operation. It reveals the need for the system to react faster to the environment. This research suggests the idea of adaptive and reactive learning and proposes the new approach of training the system with both learning threads online. The two threads have different learning rate controls and learning targets. Extensive simulations have been presented to support the fitness of the new learning system and a thorough set of measurements

have been used to understand the using-while-learning operation of the system. After concluding the learning investigation with a successful human-computer interactive experiment, a much more difficult scenario than the previous simulations, the thesis goes on to discuss issues related to implementing the proposed behaviour in an embodied mobile robot. Chapter 7 targets most commonly encountered problems when designing a mobile robot control system: the hardware and sensors, the interfaces, and the software system. The kinematics of the robot have also been analysed and experiments have been presented using a Pioneer 2DX robot to perform a learnt social positioning behaviour.

The research has been presented in two international conferences (Wang and Lee, 2006a, 2006b). This chapter aims to summarise our work and to highlight its contributions and possible generalisations. Section 8.1 describes our major contributions to knowledge and summarises them into a more general framework. Section 8.2 discusses other possible applications that can benefit from our work. Section 8.3 lays out the further possible research directions that our work can follow.

8.1 Summary of Contributions: The Using-While-Learning Framework

A summary of the major contribution of this thesis can be stated as the proposal of a novel explorative using-while-learning framework for a mobile robot social positioning behaviour in completing a human following task. This framework is set up to meet the following features of a social positioning behaviour:

Explorative learning of an unknown target

The system has to work in an explorative manner. The target function to which the system is expected to adapt is unknown beforehand, which prevents pre-training. The information of the target function can only be collected while the system is operating and learning. The features of the target function can only be revealed if the system has explored certain parts of the environment, and the system faces the challenge of learning newly explored features during the online learning operation.

No trainable error measurements

There are no desired outputs of the system thus the error performance can't be measured. The only information available is a performance index that ranks how good the system performance is. This means the error measurements-based learning strategy can not be used directly.

Need for generalisation:

It is impossible for the system to experience all possible states in a short time of operation. In order to function reasonably rather than rely on random outputs, it has to have the ability to generate reasonable outputs based on learnt knowledge. That is the ability to generalise.

The proposed using-while-learning framework is based on a multilayered feedforward neural network using backpropagation learning. The framework has three main components:

The training data selection

Because there is a lack of trainable data and the information available online is not error performance but a performance index, we proposed this method in Chapter 5 to enable the framework to convert online information into training data during the operation. This is based on recording the history of the system operations that achieve good feedback and interpreting the operations that receive negative feedback. This method improves the quality of the training data during the learning so that the learning outcome is improved. This component enables the online learning using multilayered feedforward neural networks.

Adaptive and reactive learning

The multilayered feedforward neural networks use backpropagation with two learning threads operating: the adaptive learning and the reactive learning. The adaptive learning learns all the training data and reactive learning learns only new training data in a given time window. Both learning rates are controlled by the system's error performance and the error of reactive learning is

weighted by time. This component learns and ensures that the online learning outcome is usable.

System measurements

Because the training data set keeps updating during the operation, error measurement is not enough to evaluate the system's performance. A full set of standards that can provide assessment of the system's ability have been introduced. This component provides an effective method to monitor and evaluate the online the learning operation.

The main properties and advantages of this framework include:

1. The training data selection and the learning enhance each other during operation, which is the foundation of the framework's ability to work online. It enables the error backpropagation learning that gives the social behaviour the ability to generalise.
2. The adaptive learning provides the advantage of good generalisation and convergence, while the reactive learning provides the advantage of a fast learning speed, so that the system is truly using-while-learning.
3. The two learning threads switch the control of the network dynamically so that reactive learning dominates when the system encounters new features. When the system faces no new features, as the reactive learning is not strongly active, adaptive learning has the control of the system and tunes the performance.

This using-while-learning framework forms the core of this thesis and is the main outcome of this research. The analysis provided though Chapters 5 and 6 and their comparison to Chapter 4 has clearly demonstrated the advantages of adopting this using-while-learning framework in the social positioning behaviour. However, from a higher perspective, the social positioning represents a large group of systems for which, previously, researchers found it difficult to construct a feasible using-while-learning controller. Introducing this using-while-learning framework opens a huge possibility for many other problems that remain highly motivated. In the next section, therefore, we will discuss the generalisation of this framework and explore its possible application to other research problems.

8.2 The Generalisation of the Research and Possible Applications

Although this thesis is built upon the scenario of social positioning, a wide range of applications in HRI or in general control systems can benefit from the proposed using-while-learning framework. It is easy to see that the two learning threads can be used in any learning system that uses multilayered feedforward neural networks as long as the training data are fed online. But, most importantly, by describing the training data selection method in a more general form, many well studied problems can have a promising alternative because a lack of desired system output is a common feature in many of these applications.

8.2.1 General Description of the Using-while-learning Framework

The training data selection method is the foundation of this using-while-learning framework. Here we revisit the method that was presented in Chapter 5 so that the basic requirement of adopting this method can be revealed. Chapter 5 points out that if $\mathbf{p}$ is the robot position and $\tilde{\mu}(.)$ is the system function, the system has following state transition:

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \tilde{\mu}(\mathbf{p}^k) \quad . \tag{8-1}$$

The reward function is $r(\mathbf{p}^k)$. Define $\Delta R^k = r(\mathbf{p}^{k+1}) - r(\mathbf{p}^k)$. Let $\hat{\mathcal{P}}$ be the quantised space of $\mathcal{P}$, the system state space, and $\mathcal{A}$ be the system action/output space. Also define $\Delta r_t(\hat{\mathbf{p}}) : \hat{\mathcal{P}} \to [0,1]$ to be the best reward measurement collected by making a move at state $\hat{\mathbf{p}}$, and $\mu_t(\hat{\mathbf{p}}) : \hat{\mathcal{P}} \to \mathcal{A}$ to be the action that acquired that reward measurement. Initially, $\Delta r_t(\hat{\mathbf{p}}) = 0,\ \mu_t(\hat{\mathbf{p}}) = [0,0]^{\mathrm{T}} : \forall \hat{\mathbf{p}} \in \mathcal{P}$. After data are collected, a trainable input-target data pattern is formed as $\{\hat{\mathbf{p}},\ \mu_t(\hat{\mathbf{p}})\}$. The system selects the training data as follows:

if $\Delta R^k > \Delta r_t(\hat{\mathbf{p}}^k)$:

$$\begin{aligned} \mu_t(\hat{\mathbf{p}}^k) &= \tilde{\mu}(\mathbf{p}^k) \\ \Delta r_t(\hat{\mathbf{p}}^k) &= \Delta R^k \end{aligned} \quad , \tag{8-2}$$

where $\hat{\mathbf{p}}^k \in \hat{\mathcal{P}}$ is the closest quantised state to position $\mathbf{p}^k$.

if $-1 \times \Delta R^k > \Delta r_t(\hat{\mathbf{p}}^{k+1})$:

$$\mu_t(\hat{\mathbf{p}}^{k+1}) = -1 \times \tilde{\mu}(\mathbf{p}^k) \qquad (8\text{-}3)$$
$$\Delta r_t(\hat{\mathbf{p}}^{k+1}) = -1 \times \Delta R^k$$

The primary rule (8-2) is recording the history of successful operations achieving positive feedback and the secondary rule (8-3) is interpreting the operations with negative feedback into trainable patterns. Both play a significant role in the learning.

Now, consider a general control system has the control value defined on space $\mathcal{Y}$, for which we can define a state space $\mathcal{X}$ such that:

$$\dot{\mathbf{x}} = g(\mathbf{y}, \mathbf{x}), \ \mathbf{x} \in \mathcal{X} \ \ \mathbf{y} \in \mathcal{Y} \quad . \qquad (8\text{-}4)$$

The state transition can be written in the discrete form:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + g(\mathbf{y}, \mathbf{x}) \quad . \qquad (8\text{-}5)$$

Then the training data selection method is applicable if a reward function $r : \mathcal{X} \to \mathcal{M}, \ \mathcal{M} \subset \Re$ and the function $\mathbf{y} = h(\mathbf{x}, \dot{\mathbf{x}})$ are known. $g(.)$, however, does not need to be known to execute the training data selection routine. The training data selection rule is:

if $\Delta R^k > \Delta r_t(\hat{\mathbf{x}}^k)$:

$$\mu_t(\hat{\mathbf{x}}^k) = h(\mathbf{x}^k, \dot{\mathbf{x}}|_{\mathbf{x}=\mathbf{x}^k}) = \mathbf{y}^k \qquad (8\text{-}6)$$
$$\Delta r_t(\hat{\mathbf{x}}^k) = \Delta R^k \quad ,$$

where $\hat{\mathbf{x}}^k$ is the closest quantised state to state $\mathbf{x}^k$.

if $-1 \times \Delta R^k > \Delta r_t(\hat{\mathbf{x}}^{k+1})$:

$$\mu_t(\hat{\mathbf{x}}^{k+1}) = h(\mathbf{x}^{k+1}, -\dot{\mathbf{x}}|_{\mathbf{x}=\mathbf{x}^{k+1}}) \approx h(\mathbf{x}^{k+1}, \mathbf{x}^k - \mathbf{x}^{k+1}) \qquad (8\text{-}7)$$
$$\Delta r_t(\hat{\mathbf{x}}^{k+1}) = -1 \times \Delta R^k \quad .$$

In the social positioning system, $\mathbf{y} = \tilde{\mu}(\mathbf{p})$, $\mathbf{x} = \mathbf{p}$, $h(\mathbf{x}, \dot{\mathbf{x}}) = \dot{\mathbf{x}} = \dot{\mathbf{p}} = \mathbf{y}$. For any system that fits into this model, the aim of the learning system is to estimate the relation $\mu : \mathcal{X} \to \mathcal{Y}$ so that the best reward value from the reward function can be achieved by following the steepest gradient defined on the reward surface. The peak of the reward surface is the desired state for the system to converge and stabilise on. For many systems, the desired system output is unknown but a reward function over the system states is much easier to find. As the reward function is defined upon system states that have to be explored during operation, the system can also benefit from online operation of two learning threads.

8.2.2 Possible Applications

8.2.2.1 A HRI Application: Passing Human

Robot positioning has become an attractive and interesting field in HRI, where researchers have realised that early positioning methods such as conventional obstacle avoidance (e.g. Laue and Röfer, 2005, Ge and Cui, 2000, Koren and Borenstein, 1991) and navigation (e.g. Jetto et al., 1997, Demirli and TRurksen, 2000, Aitkenhead and McDonald, 2004) are not enough because social positioning needs to take the human factor into account. This thesis has identified the limitations of using predefined human preference models in the social positioning behaviour. Such limitations largely apply to other positioning behaviour in HRI as well. One typical positioning behaviour that has been widely researched is the behaviour of passing a human. For the majority of the literature in this topic, one psychological model has been commonly used, as has been reviewed in Chapter 2 and 3.

Although social scientists have had strong evidence to support the general shape of the reward surface of a human while someone is passing them, the size and the shape of it remain highly variable among individuals. As listed in our previous studies in Chapter 2, Figure 8-1 is the most widely used preference model for passing a human.

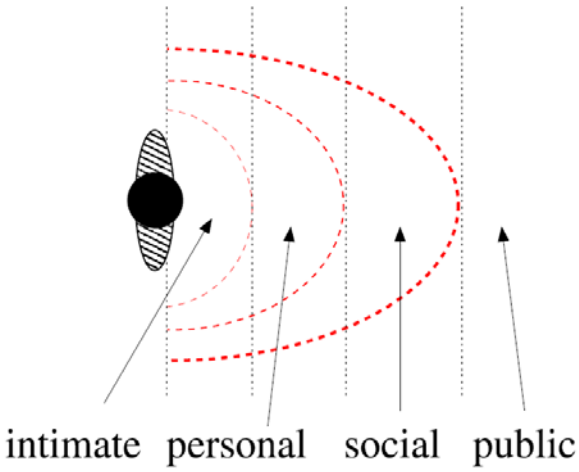


Figure 8-1 Human preference of spatial distance zones in the scenario of a robot passing a human. It generally describes how the human feels about an agent regarding the distance and position between them (Pacchierotti et al., 2006)

The desired position for passing a human will depend on both the general human preference and the dimensions of the environment. Pacchierotti (2006) has researched different passing distances in different situations. Thus the methods

proposed in previous literature (as studied in Chapter 2) lack the generality to offer an adaptive solution so that the robot can freely operate in a dynamic environment with different humans. The using-while-learning framework proposed in this thesis can benefit the passing human behaviour in a similar way to the social positioning behaviour.

As a positioning problem, it has a very similar structure to the social positioning system. The only difference will be the shape of the reward surface. As the learning system is independent of the reward surface, the using-while-learning framework can be applied to passing human behaviour directly in the same way it has been executed in social positioning system. The difference, depending on the scenario, can be that the outcome of the passing human behaviour is an aggregate of the preference of many people whereas the social positioning is defined by one user.

8.2.2.2 A Control Application: Pole Balancing

In Section 8.1, the using-while-learning framework has been presented in a more general form. This allows our work to be applied not only in position-related HRI applications but many general adaptive control applications. A well studied control problem is pole balancing. Being a commonly used model, it has been well studied within the area of conventional control methods as well as some offline based learning methods (e.g. Pasemann, 1997, Riedmiller, 2005, Atkeson, 1994, Doya, 1996). However, employing a using-while-learning controller to balance a pole remains a less studied aspect. Our work has the potential to fill this gap. Using balancing one pole with a cart as an example, the kinematicsare as follows:

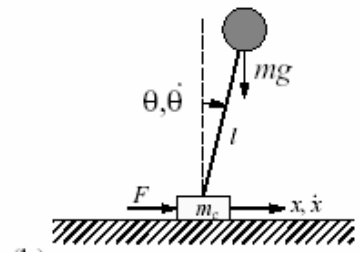


Figure 8-2 The kinematics of balancing a pole by a cart (Schaal, 1997)

Define $\mathbf{x} = [\theta, \dot{\theta}, \dot{x}]^{\mathrm{T}}$. In the pole balancing case, we can define function $g(.)$ that satisfies:

$$\dot{\mathbf{x}} = g(F, \mathbf{x}) \quad . \quad (8\text{-}8)$$

From the discussion in Section 8.2.1, it can be seen that $g(.)$ does not to be known for training data selection. All we need is to find $h(.)$. It is known that the system in Figure 8-2 satisfies following function (Ogata, 1996):

$$h(\mathbf{x}, \dot{\mathbf{x}}) = F = (m + m_c)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2 \sin\theta \,. \quad (8\text{-}9)$$

The target is to position the system in to the goal state $\mathbf{x} = [0,0,0]^T$. Adapted from the reinforcement learning reward used by Schaal (1997), the reward function can be a weighted distance from the system states to the goal state:

$$r(\mathbf{x}) = 1 - \sqrt{\frac{12\theta^2 + 0.25\dot{\theta}^2 + 1.25\dot{x}^2}{13.5}} \quad . \quad (8\text{-}10)$$

The aim of the online controller thus is to estimate the desired function $\mu(\mathbf{x}) = F$ so that the pole is balanced, i.e. the system remains in the top rewarded state, following the steepest gradient of Function (8-10). The reward function (8-10) allows the training data to be updated through the training data collection rule (8-2). Because the state transition (8-8) exists, Equation (8-9) allows the secondary rule of training data selection, rule (8-3), to be applied. Thus the using-while-learning framework has a promising potential to solve the online control of the pole balancing problem.

8.3 Future Research Directions

The using-while-learning framework has provided a promising solution for online explorative learning problems, which form a large part of the growing interest in adaptive systems. It has successfully solved the social positioning problem in this thesis. Furthermore it opens some attractive directions for future research.

8.3.1 The Online Learning Study

The proposal to use both adaptive and reactive learning is largely applicable in real-time learning where more data becomes available as learning progresses. This proposal is motivated by the high learning speed requirement in a real-time learning system where using-while-learning is necessary. Noticing that the learning algorithm is based on classic error backpropagation, it would be useful to investigate whether this learning can benefit from other learning acceleration techniques proposed in the literature. Although most of the methods proposed in existing research operate offline

and show their advantages in long-term training, using two learning threads is able to enhance their effects and has the possibility to make them have an early impact on learning.

This can open many research topics, such as, adding the popular momentum (e.g. Naimin Zhang et al., 2006) or high order error gradient backpropagation (e.g. Battiti, 1992) into the learning. These two methods have been widely used in many learning applications but their effects with two learning threads in a using-while-learning scenario are yet to be revealed. However, it has caught our attention that both methods mentioned here as well as many other discussions of the acceleration of backpropagation learning aim to avoid error local minima. The risk of experiencing error local minima in the proposed using-while-learning-frame hasn't been investigated in this thesis, although the simulations weren't observed to suffer from this problem. One of our preliminary discussions (Wang and Lee, 2006b) has pointed out that, because two learning threads have different learning targets, they experience different local minima. This may significantly reduce the risk of the system being trapped into an error local minimum as this needs the two learning threads to have the same minimum, which rarely occurs other than at the global minimum. The system's potential risk of suffering local minima and the feasibility of enhancing the system to avoid error local minima are interesting and attractive future research topics.

8.3.2 The Generalisation Study

In Section 8.2.1 we concluded that the using-while-learning framework is applicable if the following conditions are true:

- a control system can define a state transition in the form of equation (8-4) or (8-5);
- some reward function can be defined over the state space, typically the weighted distance to the goal state as equation (8-10);
- $h(.)$ is known.

It is essential to use the state transition (8-5) and $h(.)$ so that the training data collection rule (8-7) stands. The state transition must be defined so that if an action of the system leads to a state shift having negative effects on the reward, the opposite action can be found and used as training data. Our study in Chapter 5 and 6 has shown that both training data collection rule have contributed to the learning of the system.

However, it is obvious that for any system with a reward function defined on the state space, rule (8-6) is always true because it is simply a recording of system history without interpretation. Thus for systems either that are hard to linearise or that have an unknown $h(.)$ the system can still collect training data through rule (8-6). If this rule gives enough training data, the using-while-learning framework still works. Therefore, an attractive direction of research will be to find possible ways of modelling the system so that the system's dependency on the interpretation rule is reduced or minimised. Many directions are worth looking into for this purpose, such as deferent ways of modelling the system's state transition or different ways of define reward surface. Or, from another perspective, a faster learning that can train the system effectively with fewer data, where even current model can work without the interpretation rule.

As an online learning system that is independent from its learning target, the using-while-learning frame has a promising potential to be expanded into a wide range of practical and theoretical problems that provides only limited desired system target and requires strict online operation.

References

ACKLEY, D. H. & LITTMAN, M. L. (1990) Generalization and scaling in reinforcement learning. *Advances in Neural Information Processing Systems*. 2, 550-557

AHMED, R. S., RATTAN, K. S. & KHALIFA, I. H. (1995) Real-time Tracking Control of A DC Motor Using a Neural Network.

AITKENHEAD, M. J. & MCDONALD, A. J. S. (2004) Complex environments, complex behaviour. *Engineering Applications of Artificial Intelligence,* 17**,** 611-621.

ARKIN, R. C. (1998) *Behavior-Based Robotics*, The MIT Press.

ARULAMPALAM, M. S., MASKELL, S., GORDON, N. & CLAPP, T. (2002) A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE TRANSACTIONS ON SIGNAL PROCESSING,* 50.

ASADA, H. & LIU, S. (1991) Transfer of human skills to neural net robot controllers. *1991 IEEE International Conference on Robotics and Automation.* Sacramento, CA, USA.

ASHWORTH, P. D. (1979) *Social interaction and consciousness*, John Wiley & Sons.

ATKESON, C. G. (1994) Using local trajectory optimizers to speed up global optimization in dynamic programming. in MOODY, HANSON & LIPPMANN (Eds.) *Advance in Neural Information Processing Systems 6.* Morgan Kaufmann.

ATKESON, C. G. & SCHAAL, S. (1997) Robot learning from demonstration. *International Conference on Machine Learning.*

BAKKER, P. & KUNIYOSHI, Y. (1996) Robot see, robot do: An overview of robot imitation. *AIS96 Workshop on Learning in Robots and Animals.*

BATTITI, R. (1992) First- and second-order methods for learning: between steepest descent and Newton's method. *Neural Computation* 4**,** 45.

BEBIS, G. & GEORGIOPOULOS, M. (1994) Feed-forward neural networks. *IEEE Potentials.*

BEETZ, M., ARBUCKLE, T., BELKER, T., CREMERS, A. B., SCHULZ, D., BENNEWITZ, M., BURGARD, W., HÄHNEL, D., FOX, D. & GROSSKREUTZ, H. (2001) Integrated, plan-based control of autonomous robots in human environments. *IEEE Intelligent System,* 15**,** 55-65.

BELLMAN, R. (1957) *Dynamic Programming,* Princeton, NJ., Princeton University Press.

BERRY, D. A. & FRISTEDT, B. (1985) *Bandit Problems: Sequential Allocation of Experiments.,* London, UK, Chapman and Hall.

BLANCHARD, A. & CAÑAMERO, L. (2005) From imprinting to adaptation: building a history of affective interaction. *Fifth International Workshop on Epigenetic Robotics (EPIROB05).* Nara Japan.

BLUMBERG, B., DOWNIE, M., IVANOV, Y., BERLIN, M., JOHSON, M. & TOMLINSON, B. (2002) Integrated learning for interactive synthetic characters. *the ACM SIGGRAPH.*

BONABEAU, E., DORIGO, M. & THERAULAZ, G. (1999) *Swarm Intelligence: From Natural to Artificial Systems,* New York, Oxford University Press.

BOYAN, J. A. & MOORE, A. W. (1995) Generalization in Reinforcement Learning: Safely Approximating the Value Function. *Advances in Neural Information Processing Systems*. 7.

BREAZEAL, C. (2000) Believability and readability of robot faces. *The 8th International Symposium on Intelligent Robotics Systems.*

BREAZEAL, C. (2003) Towards sociable robots. *Robotics and Autonomous Systems,* 42.

BREAZEAL, C. (2004) Social interactions in HRI: The robot view. *SYSTEMS, MAN, AND CYBERNETICS, IEEE TRANSACTIONS ON,* 34**,** 181-186.

BRYSON, A. E. & HO, Y.-C. (1969) *Applied Optimal Control,* New York.

BUENO, J. I. & KRAGIC, D. (2006) Integration of tracking and adaptive gaussian mixture models for posture recognition. *The 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN06).* Hatfield, UK.

CAIRNS, R. B. (1979) Social interactional methods: An introduction. IN CAIRNS, R. B. (Ed.) *The Analysis of Social Interactions: Methods, Issues, and Illustrations.* Hillsdel, Lawrence Erlbaum Associates, Publishers**,** 1-9.

CALINON, S. & BILLARD, A. (2006) Teaching a humanoid robot to recognize and reproduce social cues. *The 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN06).* Hatfield, UK.

CANAMERO, L. & FREDSLUND, J. (2001) I show you how I like you - can you read it in my face? *IEEE Transactions on Systems, Man and Cybernetics, Part A,* 31**,** 454-459.

CHEN, D., YANG, J. & WACLTAR, H. D. (2004) Towards automatic analysis of social interaction patterns in a nursing home environment from video. *6th*

ACM SICMM International Workshop on Multimedia Information Retrieval. New York.

CHEN, F.-C. (1990) Back-propagation Neural networks for Nonlinear Self-tuning Adaptive Control. *IEEE Control Systems Magazine.*

DAUTENHAHN, K. (1997) I could be you - the phenomenological dimension of social understanding. *Cybernetics and Systems,* 28**,** 417-453.

DAUTENHAHN, K. (1999) Design spaces and niche spaces of believable social robots. *11th IEEE International Workshop on Robot and Human Interactive Communication.*

DAUTENHAHN, K. (2003) Roles and functions of robots in human society - implications from research in autism therapy. *Robotica,* 21**,** 443-452

DAUTENHAHN, K. (2004) Robots we like to live with?!-A development perspective on a personalized, life-long robot companion. *2004 IEEE International Workshop on Robot and Human Interactive Communication.* Kurashiki, Okayama Japan, University of Hertfordshire.

DAUTENHAHN, K. & BILLARD, A. (1999) Bringing up robots or - the psychology of socially intelligent robots: from theory to implementation. *The third annual conference on Autonomous Agents* Seattle, Washington, United States

DAUTENHAHN, K., WALTERS, M. L., SISBOT, E. A. & ALAMI, R. (2006) How May I Serve You? A Robot Companion Approaching a Seated Person in a Helping Context. *HRI' 06.* Salt Lake City, Utah, USA.

DEMIRLI, K. & TRURKSEN, P. B. (2000) Sonar based mobile robot localization by using fuzzy triangulation. *Robotics and Autonomous Systems***,** 109–123.

DENEUBOURG, J. L., GOSS, S., FRANKS, N., SENDOVA-FRANKS, A., DETRAIN, C. & CHRÉTIEN, L. (2000) The dynamic of collective sorting robot-like ants and ant-like robots. in MEYER, J.-A. & WILSON, S. W. (Eds.) *The First International Conference on the Simulation of Adaptive Behaviour.* MIT Press.

DOYA, K. (1996) Temporal difference learning in continuous time and space. IN TOURETZKY, D. S. & MOZER, M. C. (Eds.) *Advance in Neural Information Processing Systems 8.* MIT Press.

DUNDAR, G. & ROSE, K. (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation function. *IEEE TRANSACTIONS ON NEURAL NETWORKS,* 9**,** 224-228.

FIERRO, R. & LEWIS, F. L. (1998) Control of a nonholonomic mobile robot using neural networks. *IEEE TRANSACTIONS ON NEURAL NETWORKS,* 9**,** 589-600.

FONG, T., ILLAH, N. & DAUTENHAHN, K. (2003) A survey of socially interactive robots. *Robotics and Autonomous Systems,* 42**,** 143-166.

FORESEE, F. D. & HAGAN, M. T. (1997) Gauss-Newton approximation to bayesian learning. *International Conference on Neural Networks.*

FOX, D., BURGARDY, W., DELLAERT, F. & THRUN, S. (1999) Monte Carlo Localization: Efficient position estimation for mobile robots. *Sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence* Orlando, Florida, United States, American Association for Artificial Intelligence.

FREDA, L. & ORIOLO, G. (2007) Vision-based interception of a moving target with a nonholonomic mobile robot. *Robotics and Autonomous Systems,* 55**,** 419-432.

GAO, D. & YANG, G. (2003) Influences of variable scales and activation functions on the performances of multilayer feedforward neural networks. *Pattern Recognition,* 36**,** 869-878.

GE, S. S. & CUI, Y. J. (2000) New potential functions for mobile robot path planning. *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION,* 16.

GROSSBERG, S. (1982) *Studies of Mind and Brain,* Boston.

GRUDIC, G. Z. & LAWRENCE, P. D. (1996) Human-to-robot skill transfer using the SPORE approximation. *The 1996 IEEE International Conference on Robotics and Automation.*

HALL, E. T. (1966) *The Hidden Dimension: Men Use of Space in Public and Private,* London, The Boldley Head Ltd.

HECHT-NIELSEN, R. (1989) Theory of the backpropagation neural network. *International Joint Conference on Neural Networks.* Washington, DC, USA, .

HILGARD, E. R. & BOWER, G. H. (1975) *Theories of Learning,* Englewood Cliffs, HJ., Prentice-Hall.

HIRZINGER, G. (1996) Learning and skill acquisition. *The seventh International Symposium.* Springer, NY.

HOFFMANN, F. (2004) Fuzzy Behavior Coordination for Robot Learning from Demonstration.

HOPFILED, J. J. (1984) Neurons With Graded Response Have Collective Computational Properties Like Those of Two-state Neurons. *National Academic Science Conference.*

HORNIK, KURT, STINHCOMBE, MAXWELL, WHITE, M. & HALBERT (1988) Multilayer Feedforward Networks Are Universal Approximators.

HOWARD, R. A. (1960) *Dynamic Programming and Markov Process,* Cambridge, MA., The MIT Press.

HSIN, H. C., LI, C. C., SUN, M. & SCLABASSI, R. J. (1992) An adaptive training algorithm for back-propagation neural networks. *IEEE International Conference on System, Man, and Cybernetics.* Chicago, IL.

HUANG, D., CHEUNG, Y. & HUANG, G. (2005) A new modified hybrid learning algorithm for feedforward neural networks. *Advances in Neural Networks - Second International Symposium on Neural Networks.*

HUANG, G. B. & BABRI, H. A. (2000) Classification ability of single hidden layer feedforward neural networks. *IEEE TRANSACTIONS ON NEURAL NETWORKS,* 9**,** 4.

ISBELL, C., SHELTON, C., KEARNS, M., SINGH, S. & STONE, P. (2001) Cobot: A social reinforcement learning agent. *5th International Conference on Autonomous Agents.*

JETTO, L., LONGHI, S. & VITALI, D. (1997) Localization of a wheeled mobile robot by sensor data fusion based on a fuzzy logic adapted Kalman Filter. *Control Engineering Practice,* 7**,** 763-771.

JIANG, Z.-P. & NIJMEIJER, H. (1997) Tracking Control of Mobile Robots: A case Study in Backstepping. *Automatica,* 33**,** 1393-1399.

KAELBLING, L. P., LITTMAN, M. L. & MOORE, A. W. (1996) Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research,* 4**,** 237-285.

KANDA, T., ISHIGURO, H., IMAI, M. & ONO, T. (2003) Body Movement Analysis of Human-Robot Interaction. *International Joint Conference on Artificial Intelligence.*

KANDA, T., ISHIGURO, H., ONO, T., IMAI, M. & NAKATSU, R. (2002) Development and Evaluation of an Interactive Humanoid Robot "Robovie". *IEEE International Conference on Robotics and Automation.*

KAPLAN, F., OUDEYER, P., KUBINYI, E. & MIKLOSI, A. (2002) Robotic clicker training. *Robotics and Autonomous Systems,* 38**,** 197-206.

KASPER, M., FRICKE, G., STEUERNAGEL, K. & PUTTKAMER, E. V. (2001) A behavior-based mobile robot architecture for Learning from Demonstration. *Robotics and Autonomous Systems,* 24**,** 153-164.

KHAW, J. F. C., LIM, B. S. & LIM, L. E. N. (1995) Optimal design of neural networks using the Taguchi method. *Neurocomputing,* 7**,** 20.

KIM, Y.-S. & YUM, B.-J. (2004) Robust design of multilayer feedforward neural networks: an experimental approach. *Engineering Applications of Artificial Intelligence,* 17.

KOAY, K. L., WALTERS, M. L. & DAUTENHAHN, K. (2005) Methodological issues using a comfort level device in human-robot interactions. *The 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN06).* Hatfield, UK.

KOBAYASHI, H., HARA, F. & TANGE, A. (1994) A basic study on dynamic control of facial expressions for face robot. *The 3rd IEEE International Symposium on Robot and Human Interactive Communication.* Nagoya, Japan.

KOHONEN, T. (1982) Self-organized formation of topologically correct feature maps. *Biological Cybernetics,* 43.

KOREN, Y. & BORENSTEIN, J. (1991) Potential field methods and their inherent limitations for mobile robot navigation. *the IEEE Conference on Robotics and Automation.* Sacramento, California.

KUBE, C. & BONABEAU, E. (2000) Cooperative transport by ants and robots. *Robotics and Autonomous Systems,* 30.

LAUE, T. & RÖFER, T. (2005) A behavior architecture for autonomous mobile robots based on potential fields. *The 8th International Workshop on RoboCup 2004.* Springer, Erscheinen.

LEWIS, F. L., LIU, K. & YESILDIREK, A. (1995) Neural net robot controller with guaranteed tracking performance. *IEEE TRANSACTIONS ON NEURAL NETWORKS,* 6**,** 703-715.

MAES, P. (1993) Behavior-based artificial intelligence. IN MEYER, J. A., ROITBLAT, H. L. & WILSON, S. W. (Eds.) *Animals to Animats 2: the Second International Conference on Simulation of Adaptive Behavior.* MIT Press, Cambridge.

MALMBERG, M. (1980) *Human Territoriality: Survey of Behavioural Territories in Man with Preliminary Analysis and Discussion of Meaning*, Mouton Publishers.

MELHUISH, C., HOLLAND, O. & HODDELL, S. (1998) Collective sorting and segregation in robots with minimal sensing. IN PFEIFER, R., BLUMBERG, B., MEYER, J.-A. & WILSON, S. W. (Eds.) *the fifth international conference on simulation of adaptive behavior on From animals to animats.* MIT Press, 1998.

MENON, A., MEHROTRA, K., MOHAN, C. K. & RANKA, S. (1996) Characterization of a class of sigmoid functions with applications to neural networks. *Neural Networks,* 9**,** 26.

MITSUNAGA, N., SMITH, C., KANDA, T., ISHIGURO, H. & HAGITA, N. (2005) Robot behaviour adaptation for Human-Robot Interaction based on policy gradient reinforcement learning. *RSJ International Conference on Intelligent Robots and Systems.*

MURPHY, R. R. (2004) Human-Robot Interaction in rescue robotics. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, PART C,* 34**,** 138-153.

NAIMIN ZHANG, WEI WU & GAOFENG ZHENG (2006) Convergence of gradient method with momentum for two-layer feedforward neural networks. *IEEE TRANSACTIONS ON NEURAL NETWORKS,* 17**,** 522-525.

NAKAUCHI, Y. (2002) A social robot that stands in line. *Autonomous Robots,* 12**,** 313-324.

NARENDRA, K. S. & THATHACHAR, M. A. L. (1974) Learning automata - a survey. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS,* 4**,** 323-334.

NARENDRA, K. S. & THATHACHAR, M. A. L. (1989) *Learning Automata: An Introduction,* Englewood Cliffs, NJ., Prentice-Hall.

NICOLESCU, M. (2003) Natural methods for robot task learning: Instructive demonstrations, generalization and practice. *2nd International Conference of AAMAS.* Melbourne, Australia.

NICOLESCU, M., JENKINS, O. & OLENDERSKI, A. (2006) Learning behaviour fusion estimation from demonstration. *The 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN06).* Hatfield, UK.

NICOLESCU, M. & MATARIC, M. J. (2001) Learning and interacting in human-robot domains. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, Part A: System and Humans,* 31**,** 419-430.

OGATA, K. (1996) Modern Control Engineering. 3rd ed., Prentice-Hall, Inc.**,** p86.

PACCHIEROTTI, E., CHRISTENSEN, H. I. & JENSFELT, P. (2006) Evaluation of Passing Distance for Social Robots. *The 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN06).* Hatfiled, UK.

PASEMANN, F. (1997) Pole-Balancing with different evolved neurocontrollers. IN GERSTNER, W., GERMOND, A., HASLER, M. & NICOUD, J. D. (Eds.) *Artificial Neural Networks - ICANN'97.* Lausanne, Switzerland, Spinger-Verlag.

PATTERSON, D. W. (1996) *Artificial Neural Networks: Theory and Application,* Singapore, Simon & Schusrer (Asia) Pte Ltd.

PAULOS, E. & F., C. (2001) Designing personal tele-embodiment. *Autonomous Robots,* 11.

PUTERMAN, M. L. (1994) *Markove Decision Process - Discrete Stochastic Dynamic Programming,* New York, NY., John wiley & Son Inc.

RIEDMILLER, M. (2005) Neural reinforcement learning to swing-up and balance a real pole. *IEEE International Conference on Systems, Man and Cybernetics.*

ROY, D. K. & PENTLAND, A. (2002) Learning words from sights and sounds: a computational model. *Cognitive Science,* 26**,** 113-146.

RUMMERY, G. A. & NIRANJAN, M. (1994) On-Line Q-Learning Using Connectionist Systems Cambridge University Engineering Department.

SACK, R. (1986) *Human Territoriality,* Cambridge, UK, Cambridge University Press

SCASSELLATI, B. (2000) *Foundation for a theory of social development using a humanoid robot* Cambridge, MIT Press.

SCHAAL, S. (1997) Learning from demonstration. *Advances in Neural Information Processing System,* 9**,** 1040-1046.

SCHAAL, S. (1999) Is imitation learning the route to humanoid robots. *Trends in Cognitive Sciences,* 3**,** 233-242.

SCHEEFF, M., PINTO, J., RAHARDJA, K., SNIBBE, S. & TOW, R. (2000) Experiments with Sparky: a social robot. *Workshop of Interactive Robot Entertainment.*

SECORD, P. & BACKMAN, C. (1964) *Social Psychology,* New York, McGrawHill.

SEKMEN, A. S., WILKES, M. & KAWAMURA, K. (2002) An application of passive Human-Robot Interaction: Human tracking based on attention distraction. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, Part A: System and Humans,* 32**,** 248-259.

SIMMONS, R., GOODWIN, R., ZITA HAIGH, K., KOENIG, S. & O'SULLIVAN, J. (1997) A layered architecture for office delivery robots. *Proceedings of Autonomous Agents* 245–252.

SKUBIC, M., PERZANOWSKI, D., BLISARD, S., SCHULTZ, A., ADAMS, W., BUGAJSKA, M. & BROCK, D. (2004) Spatial language for Human-Robot Dialogs. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, Part C,* 34**,** 154-167.

SONG-YEE, Y., BRUCE, M. B. & GERALD, E. S. (2000) Motivation driven learning for interactive synthetic characters. *Proceedings of the fourth international conference on Autonomous agents.* Barcelona, Spain, ACM Press.

STEIL, J. J. (2005) Online stability of backpropagation-decorrelation recurrent learning.

STRICKLIN, W. R., ZHOU, J. Z. & GONYOU, H. W. (1995) Selfish animals and robot ethnology: using artificial animals to investigate social and spatial behavior. *Applied Animal Behaviour Science,* 44**,** 187-203.

STROEVE, S. (1998) An analysis of learning control by backpropagation though time. *Neural Networks,* 11**,** 709-721.

SUN, S. (2005) Designing approach on trajectory-tracking control of mobile robot. *Robotics and Computer-Integrated Manufacturing,* 21**,** 81-51.

SUTTON, R. S. (1988) Learning to predict by the methods of temporal difference. *Machine Learning,* 3**,** 9-44.

SUTTON, R. S. (1990) Integrated architectures of learning, planning, and reacting based on approximating dynamic programming. *The 7th International Conference on Machine Learning.* San Mateo, CA, Morgan Kaufmann.

SUTTON, R. S. (1996) Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems*, 8,1038-1044.

SYRDAL, D. S., DAUTENHAHN, K., WOODS, S., WATERS, M. L. & KOAY, K. L. (2006) 'Doing the right thing wrong' - Personally and tolerance to uncomfortable robot appearance. *The 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN06).* Hatfield, UK.

TARDÓS, J. D. & NEIRA, J. (2002) Robust mapping and localization in indoor environments using sonar data. *The International Journal of Robotics Research,*.

TEWS, A. D., SUKHATME, G. S. & MATARIC, M. J. (2002) An infrastructure for large-scale human-robot interaction. *CRES Technical Report,* 1.

THRUN, S., FOX, D., BURGARD, W. & DELLAERT, F. (2001) Robust Monte Carlo localization for mobile robots. *Artificial Intelligence,* 128**,** 99–141.

TRIESCH, J., TEUSCHER, C., DEÁK, G. O. & CARLSON, E. (2006) Gaze following: why (not) learn it? *Developmental Science,* 9**,** 125-157.

WANG, Y. & LEE, D. (2006a) Reinforcement learning for a human-following robot. *The 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN06)*. Hatfield, UK.

WANG, Y. & LEE, D. (2006b) Online Backpropagation Learning for a Human-following Mobile Robot. *Towards Autonomous Robotic Systems (TAROS 2006).* Guildford, UK.

WATKINS, C. J. (1989) Learning from delayed rewards. Cambridge, UK, University of Cambridge.

WATKINS, C. J. & DAYAN, P. (1992) Technical Note: Q-learning. *Machine Learning,* 8.

WERBOS, P. J. (1988) Backpropagation: Past and Future. *1988 IEEE International Conference on Neural Networks.* New York, IEEE Press.

WILLIAMS, R. J. (1983) Unit Activation Rules For Cognitive Network Models. San Diego, Institute of Cognitive Science, University of California at San Diego.

WOODS, S., WALTERS, M. L., KOAY, K. L. & DAUTENHAHN, K. (2006) Comparing Human-Robot Interaction scenarios using live and video based methods: Towards a novel methodological approach. *9th IEEE International Workshop on Advanced Motion Control.*

YANG, S. M. & LEE, G. S. (1999) Neural network design by using Taguchi method. *Journal of Dynamic System Measurement and Control: Transactions of ASME,* 121**,** 560-563.

YODA, M. & SHIOTA, Y. (1996) Analysis of human avoidance motion for application to robot. *Robot and Human Communication, 1996., 5th IEEE International Workshop on.*

YODA, M. & SHIOTA, Y. (1997) The mobile robot which passes a man. *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN).*

ZAVLANGAS, P. G., TZAFESTAS, S. G. & ALTHOEFER, K. (2000) Fuzzy obstacle avoidance and navigation for omnidirectional mobile robots. *ESIT.* Aachen, Germany.

ZEIDENBERG, M. (1990) *Neural Networks in Artificial Intelligence,* Chichester, Ellis Horwood Limited.

ZHAO, H., DANG, K. & LIN, T. (2002) A Online-Trained Neural Network Controller for Electro-hydraulic Servo System. *The 4th World Conference on Intelligent Control and Automation.* Shanghai, P.R. China.

Appendix I Data and Measurements of the Online Simulation in Section 5.4.1

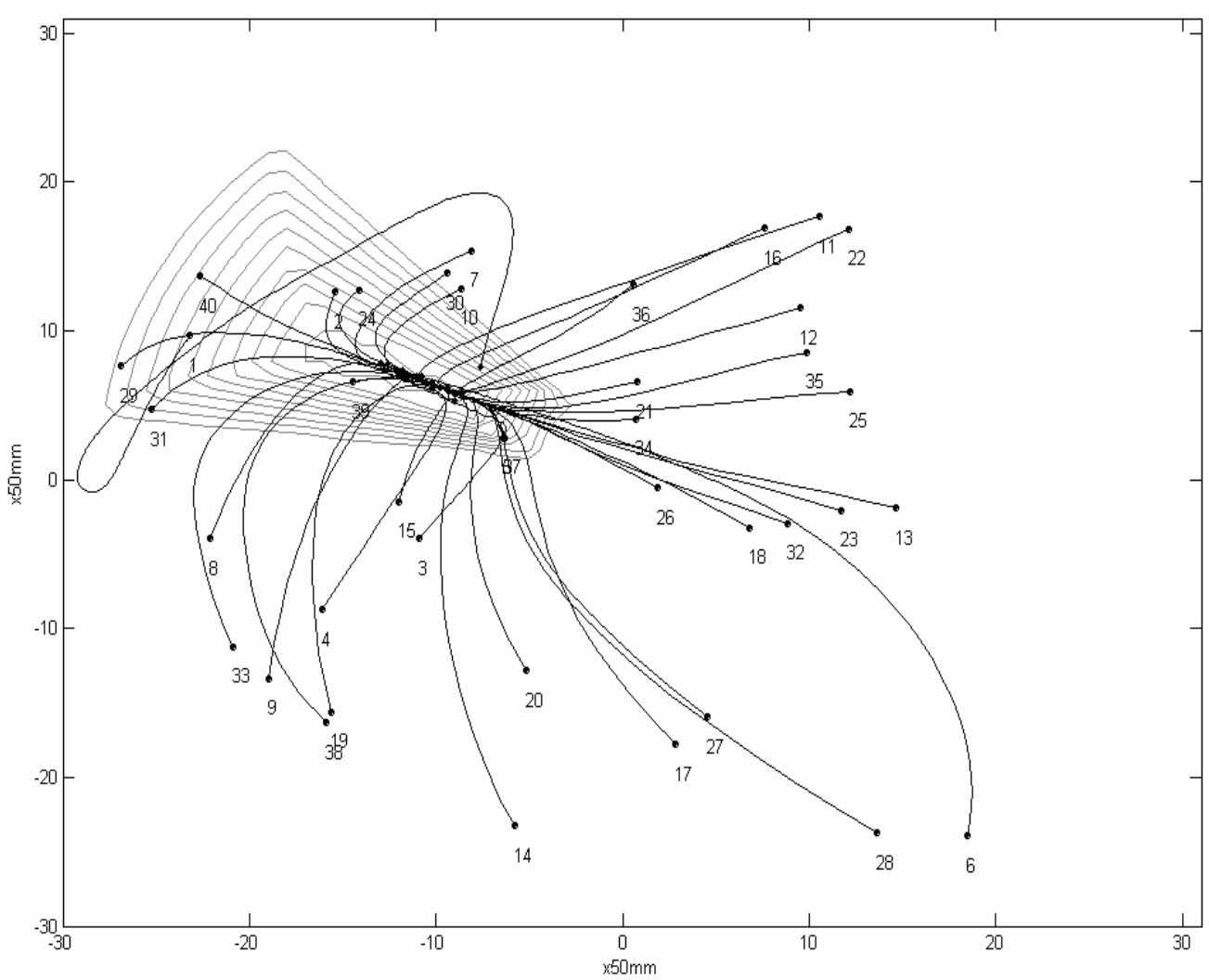


Figure I-1 All walks of the simulation of the online backpropagation learning system with reward surface having one reward peak and wide flat area

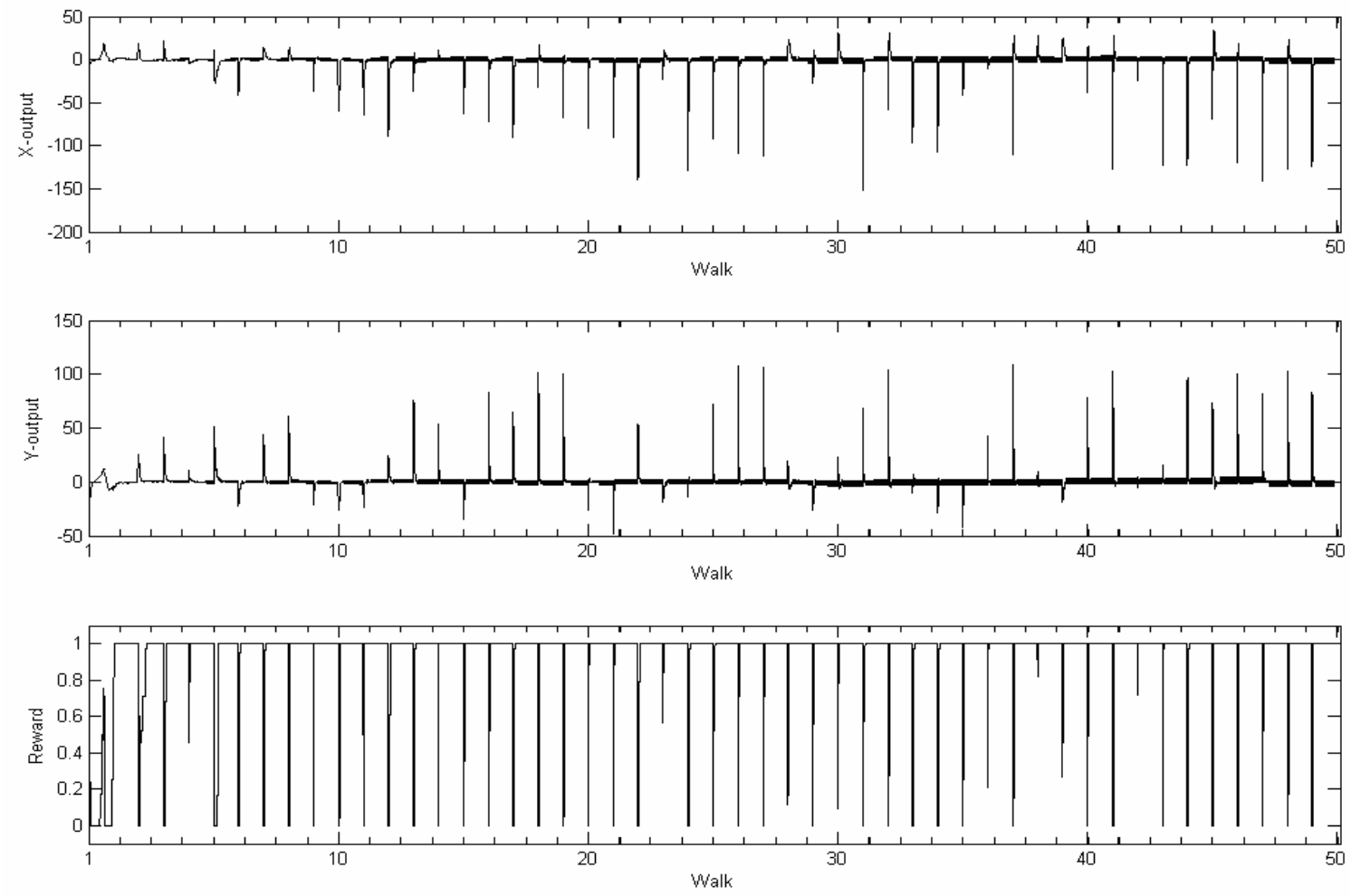


Figure I-2 The system outputs and the reward history of the simulation of the online backpropagation learning system with reward surface having one reward peak and wide flat area

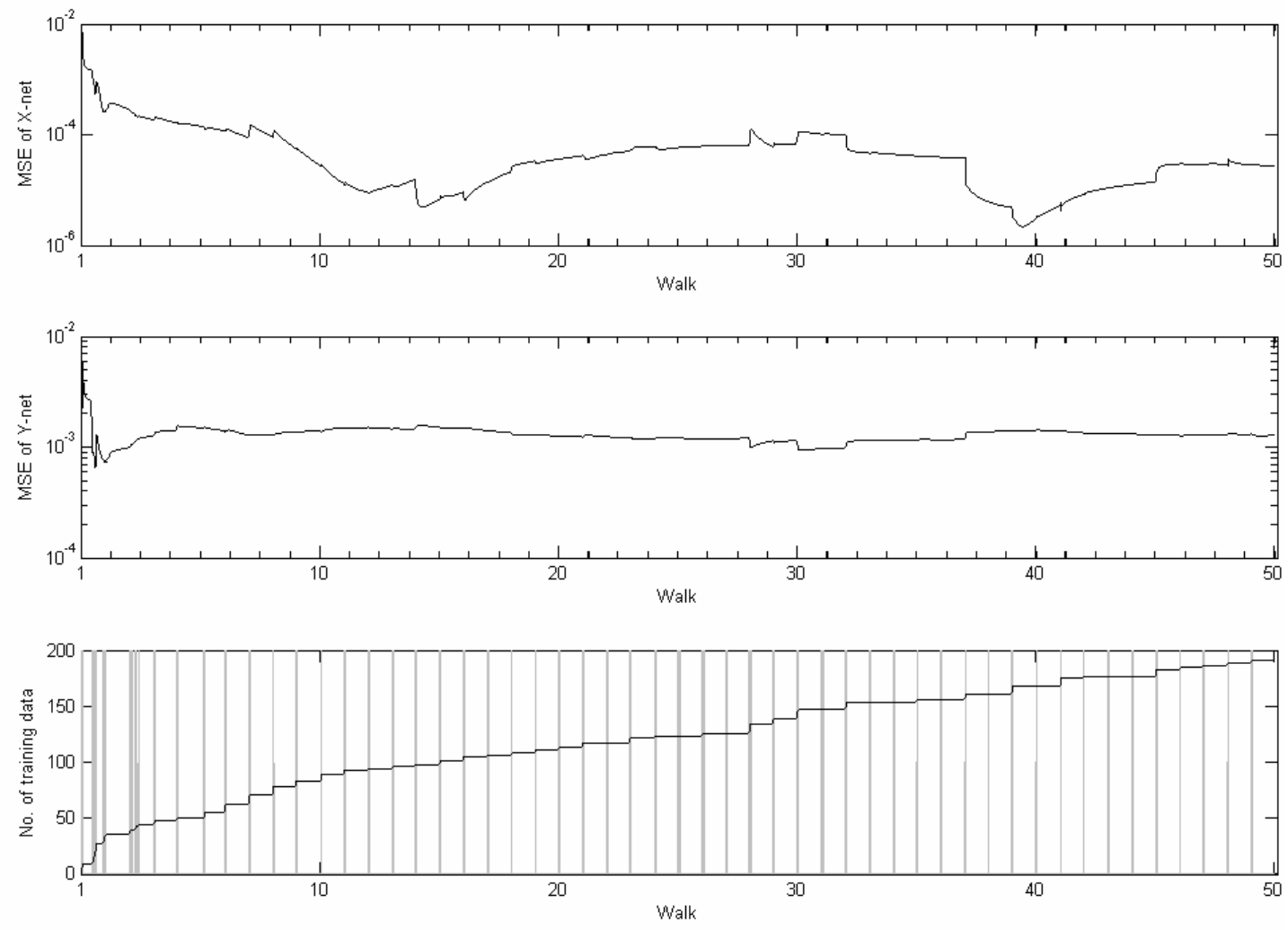


Figure I-3 The MSE and training data updates of the simulation of the online backpropagation learning system with reward surface having one reward peak and wide flat area

Appendix II Data and Measurements of the Online Simulation in Section 6.1.2

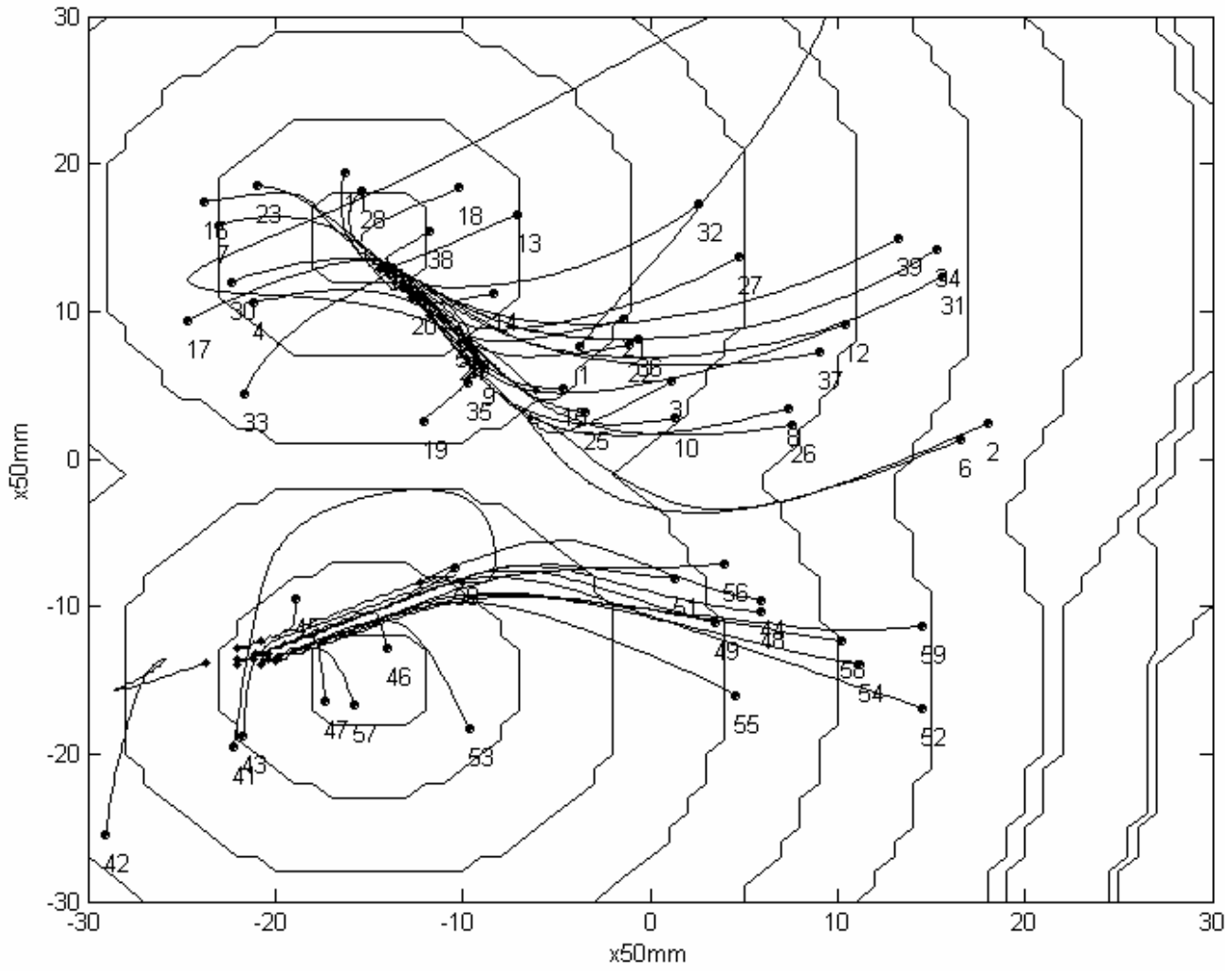


Figure II-1 All walks of the simulation of the online backpropagation learning system with reward surface having two reward peaks. The system operates in the upper half of the map for 40 walks and then operates at the lower half in 20 walks.

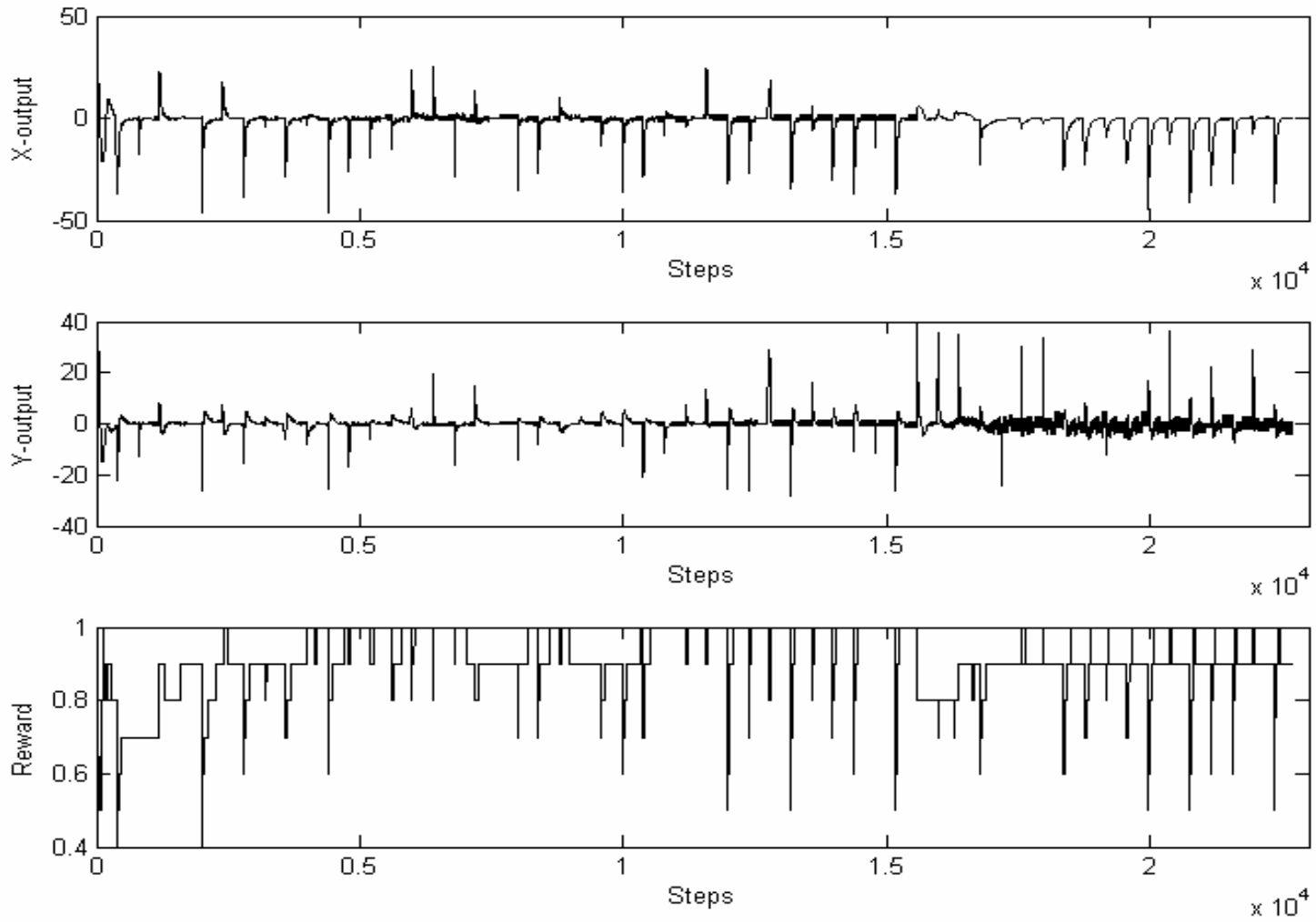


Figure II-2 The system outputs and the reward history of the simulation of the online backpropagation learning system with reward surface having one reward peak and wide flat area

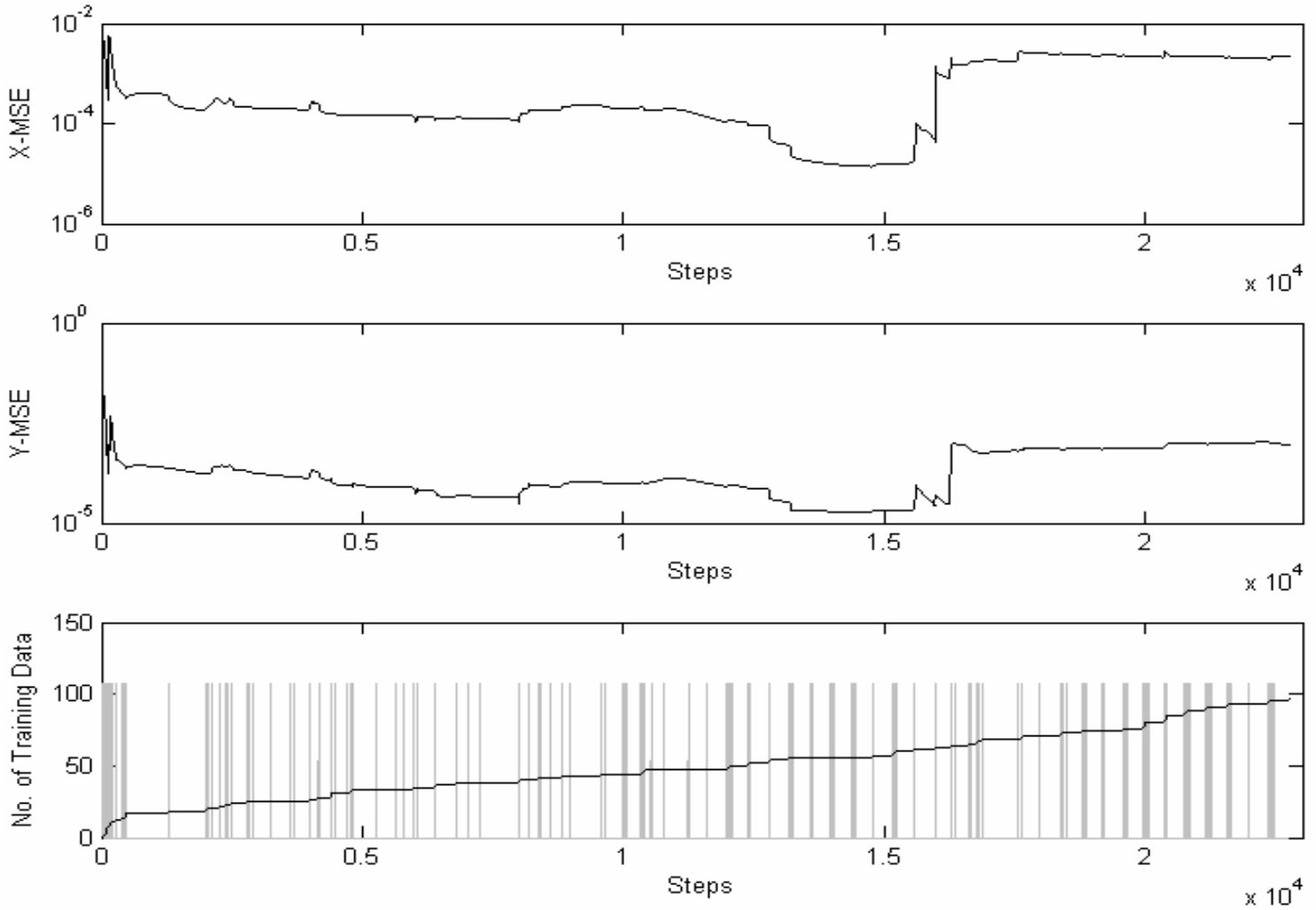


Figure II-3 The MSE and training data updates of the simulation of the online backpropagation learning system with reward surface having one reward peak and wide flat area

Appendix III Data and Measurements of the Online Simulation in Section 6.4

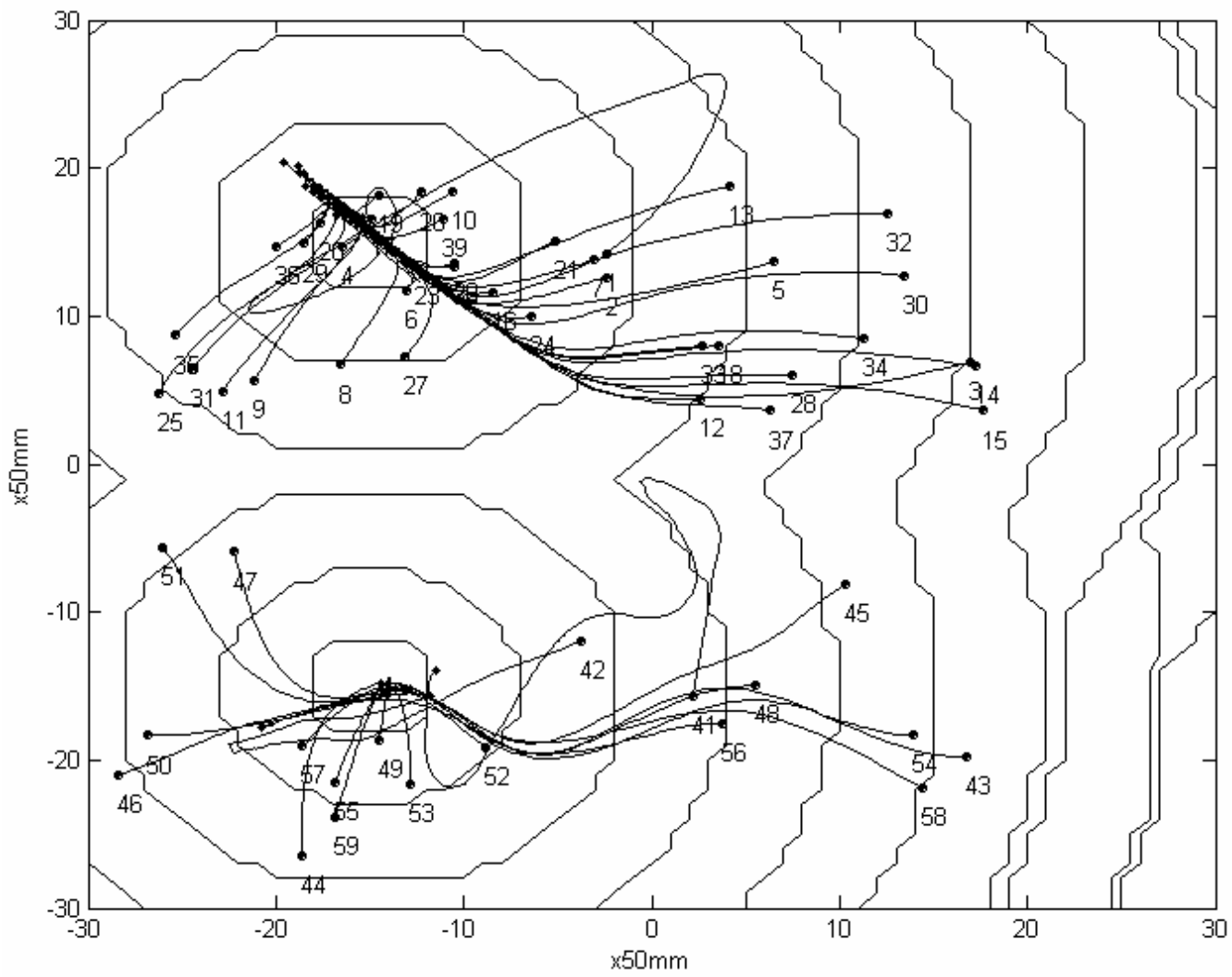


Figure III-1 All walks of the simulation of the system using two learning threads with reward surface having two reward peaks. The system operates in the upper half of the map for 40 walks and then operates at the lower half in 20 walks.

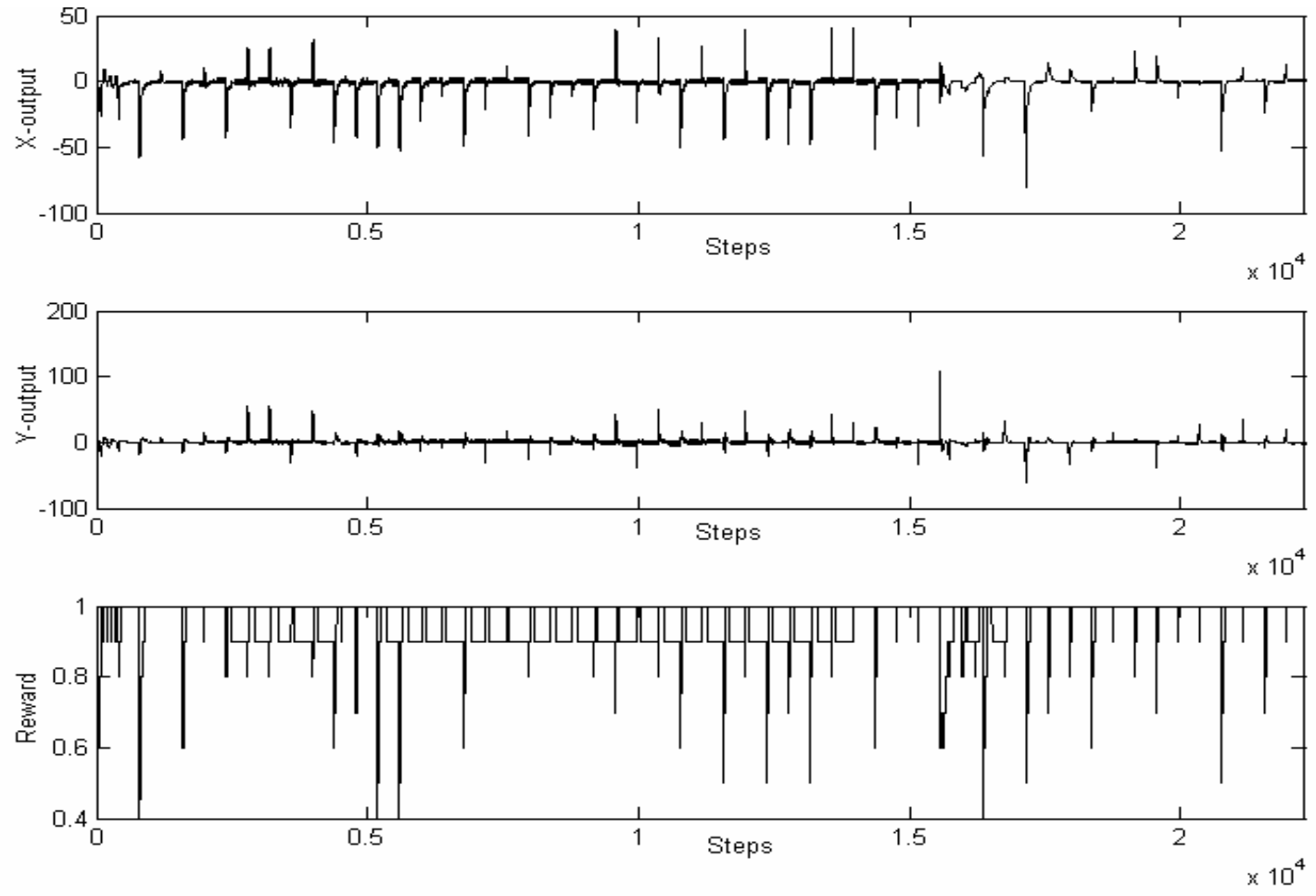


Figure III-2 The system outputs and the reward history of the simulation of the system using two learning threads with reward surface having one reward peak and wide flat area

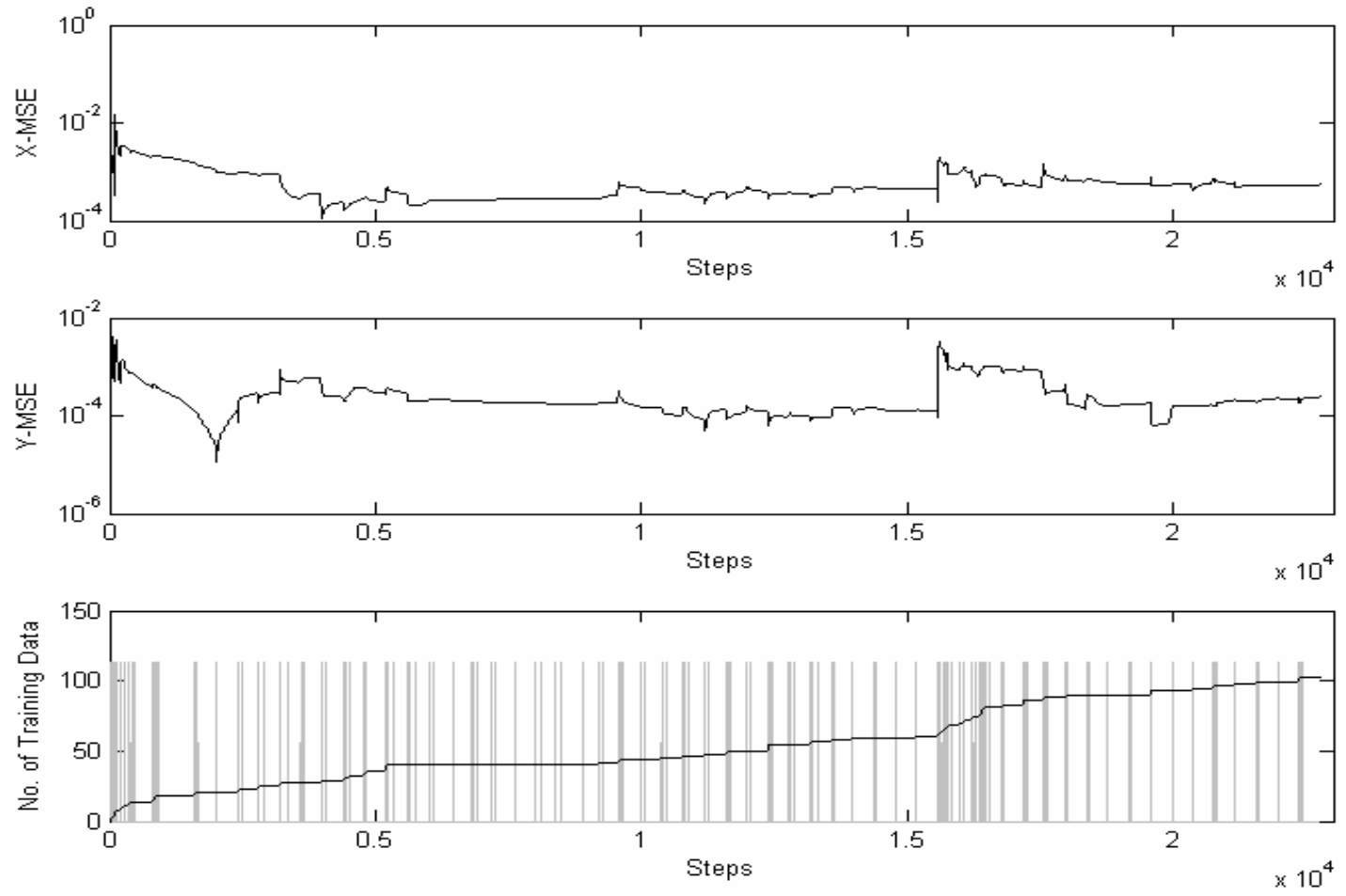


Figure III-2 The MSE and training data updates of the simulation of the system using two learning threads with reward surface having one reward peak and wide flat area

Appendix IV Onboard Controllers

There are three embedded on-board controllers provided by the manufacturer. They are: motor/power controller, sensor controller and system microcontroller.

Because the control of the DC motor wheels of the robot is a regular task involved in controlling mobile robots, a pre-built controller has been included in the micro control system for the controlling of the motor wheels. The control law of the mobile robot wheels is normally complex and has been discussed a lot in the literature (e.g. Jiang, 1997, Shima, 2003, Sun, 2005). The on-board motor controller has provided a pre-built law that enables a robust performance of the robot in low and middle speed ranges, where Section 7.4 gives more details. It has been advised by the manufacturer that users do not by-pass this controller for a robust enough performance in regular use, unless absolutely necessary.

The sonar controller manages the sonar array of the robot and provides a pre-built sonar model as well. It controls the sonar operations as well as transforms the raw sonar readings into rage readings. No extra modelling therefore is needed.

The system microcontroller is the core of the Pioneer 2 operating system. Both sonar and motor controllers are controlled by this microcontroller. It can communicate with computers allowing the robot to be supervised by external programmes. The system I/O is provided through a 9-pin DSUB serial port. The data package can be readable to computer programming languages through the interpretation of the ARIA class provided by the manufacturer. The system has reserved system I/O for optional accessories from the manufacturer such as ultrasound sonar, laser detector, robot arm or camera. For other customised accessories, they can only be connected to the system through user I/O expansion.

The user I/O expansion is a 20-pin IDC socket on the microcontroller board. It supplies 8 general-purpose digital I/O ports and 5 A/D converter ports. In our research, the human detector is connected to this user I/O using three of its A/D ports.

Appendix V Model of Human Detecting Device

The device has two detectors. The outputs of each detector include the reading of the length of the wire, d_l, and two readings from the rotational base related to the wire's spatial angles, d_x and d_y. The three outputs are connected through three A/D ports in user I/O expansion on the Pioneer 2 DX microcontroller, each of which provides a ten-bit quantisation of its signal.

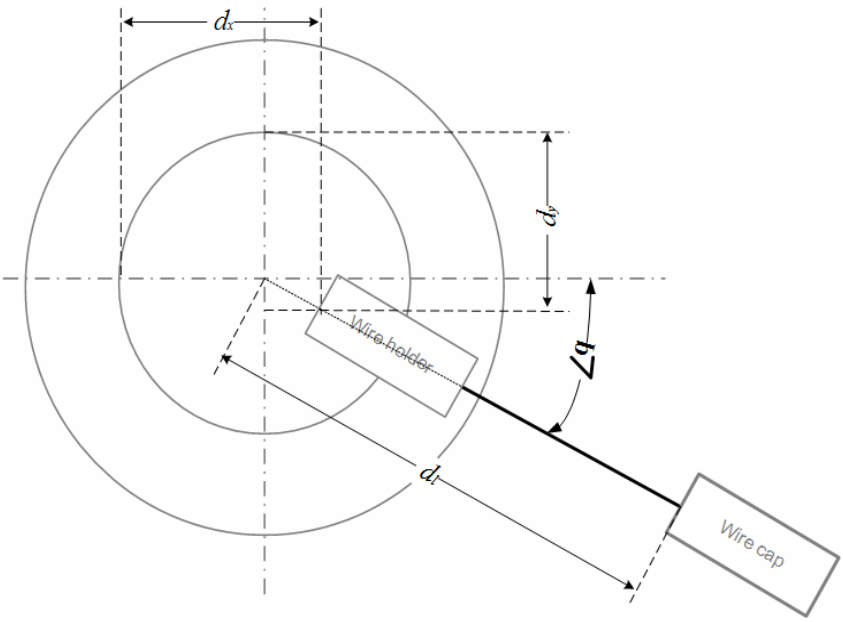


Figure V-1 GameTrak^{TM} motion senor outputs

The sensor is mounted on the deck at the top of the robot as shown in Figure 7-3. The centre of the detector is placed at the centre of rotation of the robot. The exact position of the point of attachment can be calculated through the reading of the detector. However, the GameTrak sensor only offers a limited range of freedom of the wire holder within the ring of the base holder. This means that in the majority of cases during operation, the holder will be caught up on the edge of the base holder as illustrated in Figure V-2.

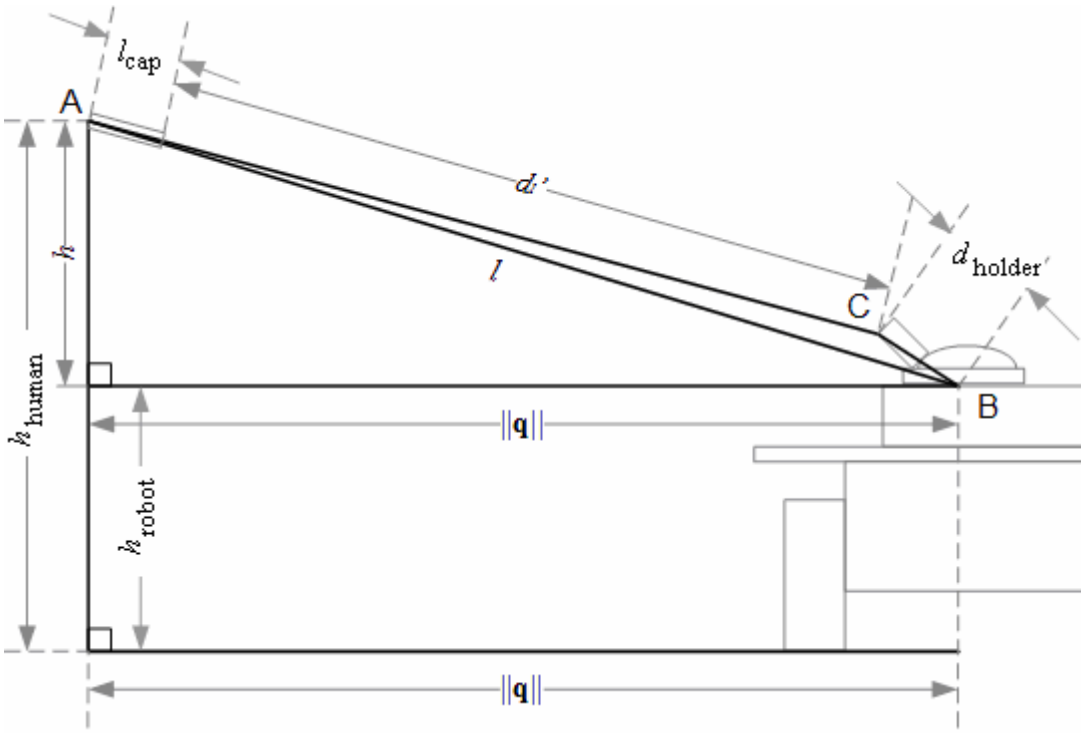


Figure V-2 Geometry of GameTrak^{TM} motion senor

Figure V-2 shows the geometry of the detector during operation in the case that the wire holder has reached the edge of the base holder. In the social positioning behaviour model, the system is studied in the x-y plane, the top-down view. Therefore, the aim here is to compute the shadow of the wire on x-y plane, which is denoted as **q**, the human position vector in robot system (see Section 3.2).

Let $\angle\mathbf{q}$ denote the angle of vector **q**. Through Figure V-2, the angle of **q** can be decided by following equation:

$$\angle\mathbf{q} = \text{atan2}(d_x\text{-}512, d_y\text{-}512) \qquad \text{(V-1)}$$

where:

$$\text{atan2}(x, y) = \begin{cases} \text{atan}(\frac{x}{y}), \text{if } y \neq 0 \\ 0, \text{if } y = 0 \end{cases} \qquad \text{(V-2)}$$

In Figure V-2, as $\angle ABC$ is expected to be very small, l is approximated as:

$$l \approx l_{\text{cap}} + g(d_l) = l_{\text{cap}} + g(d_l' + d_{\text{hold}}) \qquad \text{(V-3)}$$

where g is the transition between the measurements of the wire length and the quantised digital output of the detector. GameTrak doesn't publish its data sheet for

public use therefore the decision of the transition was determined through experiments.

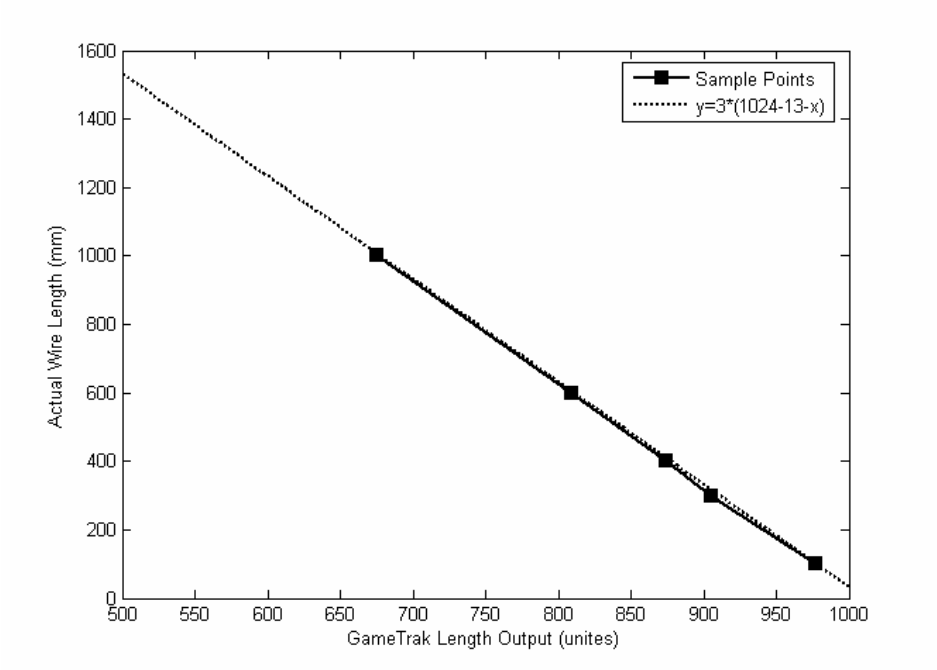


Figure V-3 The relation of the detector outputs and actual measurements in wire length

Sampling a set of wire lengths, the relation between the quantised detector length outputs and the actual wire length measurements is shown in Figure V-3. In Figure V-3, the data fit into the line y=3[(1024-x)-13]. Therefore:

$$g(x) = 3\text{mm} \times [(1024 - x) - 13] \qquad \text{(V-4)}$$

The height from the attachment point on the human to the ground, h_{human}, and the height of the robot, h_{robot}, as shown in Figure V-2, will be pre-measured. Thus, the distance from the human to the robot, ||**q**||, is:

$$\|\mathbf{q}\|^2 = l^2 - h^2 = l^2 - (h_{\text{human}} - h_{\text{robot}})^2 \qquad \text{(V-5)}$$

where h is the difference between the height of the pre-measured attachment point and the height of the robot. Therefore vector **q** is:

$$\mathbf{q} = \|\mathbf{q}\| \cdot [\cos(\angle\mathbf{q}), \sin(\angle\mathbf{q})] \qquad \text{(V-6)}$$

It has been measured that the length of the cap, l_{cap}, is 70mm, the length of the holder in quantised value, d_{holder}, is 8 and the height of the robot h_{robot} is 300mm.

Appendix VI The Kinematics Model

The variables that are available directly from the robot system are the instantaneous robot translational speed v and rotational speed ω. The human detector also gives the human position, $\mathbf{q}$, in the current robot coordinate frame. However, in order to complete the human-following task, there are three more variables which need to be deduced: the robot position in the human reference frame, $\mathbf{p}$; the human displacement $\mathbf{s}_h$ and robot displacement $\mathbf{s}_r$, for the given time window. The derivation of these variables, based on the robot kinematics, is the aim of the analysis of this section. This section will also calculate the control signal for the robot movements.

There are three frames of reference involved in the current study: the global reference frame, the robot reference frame and the human reference frame. In the robot reference frame the robot is considered to be still and always facing along the positive x axis. It is the reference frame of all sensor readings. In the human reference frame the human is considered to be still and always facing along the positive x axis. The human reference frame is where the social positioning model is based.

In the majority of the literature, kinematics are discussed in the global coordinate frame. An illustration of the positions of the robot and human at time steps k and u is shown in Figure VI-1. We use prime to denote the measurements of a vector in the global frame.

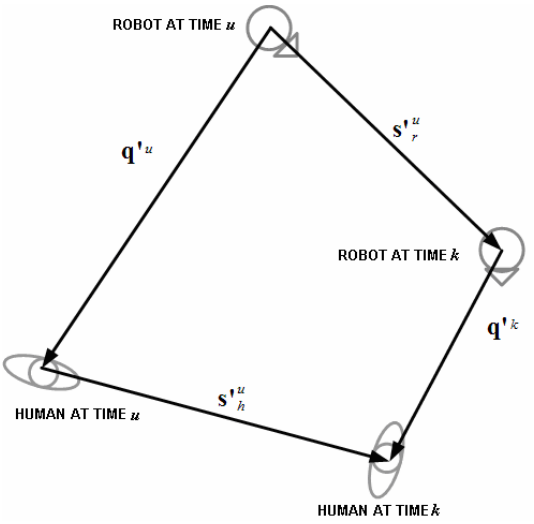


Figure VI-1 The geometry of the human-following kinematics in global frame of reference. It illustrates the relation of the positions of the human and the robot at two time instance k and u.

In Figure VI-, it is obvious that we have the following equation:

$$\mathbf{q}^u + \mathbf{s}_h^u = \mathbf{q}^k + \mathbf{s}_r^u \quad . \qquad \text{(VI-1)}$$

However, there is no global localisation facility in the system. The measurements of vectors in Equation (VI-1) are therefore unknown. Also, because the position reading of the human, **q,** is located in robot coordinate frame, and the analysis of the kinematics of the mobile robot takes place in the robot reference frame, the rotation of the robot itself has to be taken into account during the calculation of the global vectors. If we redraw the situation of Figure VI- in the frame of robot reference, Figure VI- is produced. From this point on, unless otherwise specified, all the vectors mentioned in this section are measured in the robot's coordinate frame.

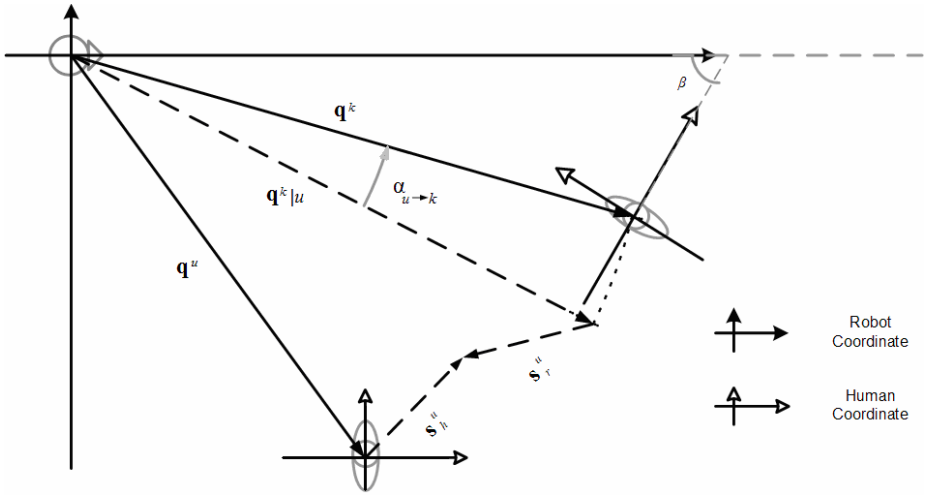


Figure VI-2 The relation of the human movements in the robot reference frame at two time indices.

As shown in Figure VI-2, the difference between the readings $\mathbf{q}^k$ and $\mathbf{q}^u$ is caused by the movements of the robot, the human and the rotation of the frame of reference from time index u to k. Because the reference system rotates during the process, the calculation of the vectors is not correct unless they are all undertaken with their observations from the same coordinate frame. We use $\mathbf{q}^u|k$ to denote the observation of vector $\mathbf{q}^u$ in at time index k in the same reference frame. For convenience, write $\mathbf{q}^u|u$ as $\mathbf{q}^u$, and $\mathbf{q}^k|k$ as $\mathbf{q}^k$. The rotation of the reference frame will have the effect of an opposite of rotation of a vector so that:

$$\mathbf{q}^k \mid k = \mathbf{q}^k = \mathbf{T}_{\alpha_{u\to k}} \times \mathbf{q}^k \mid u \quad , \qquad \text{(VI -2)}$$

where $\alpha_{u\to k}$ is the angle difference of the coordinate from time index u to k and $\mathbf{T}$ is the rotational matrix:

$$\mathbf{T}_\alpha = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \quad . \qquad \text{(VI -3)}$$

Therefore, from Figure VI-, it can be seen that: $\mathbf{q}^k \mid_u = \mathbf{q}^u + \mathbf{s}_h^u - \mathbf{s}_r^u$. Equivalently:

$$\mathbf{q}^k = (\mathbf{q}^u + \mathbf{s}_h^u - \mathbf{s}_r^u) \mid k \quad . \qquad \text{(VI-4)}$$

In order to derive the human displacement $\mathbf{s}_h$ from Equation (VI-4), the human displacement can be written as:

$$\mathbf{s}_h^u \mid k = \mathbf{q}^k - \mathbf{q}^u \mid k + \mathbf{s}_r^u \mid k \quad . \qquad \text{(VI-5)}$$

From Equation (VI-5), it can be seen that the displacement of the robot, $\mathbf{s}_r$, is needed for the calculation. The instantaneous translational and rotational speed readings are given by the robot system. The common assumption adopted in most research concerning the kinematics of a wheeled mobile robot is a unicycle movement in short time length. We use the approximation Freda and Oriolo (2007) adopted in their research. The displacement of the robot between two adjacent time indices can be approximated as following:

$$\mathbf{s}_r^{k-1 \to k} \approx \overline{v} \cdot [\cos(\frac{\overline{\omega}}{2}), \sin(\frac{\overline{\omega}}{2})] \quad , \qquad \text{(VI -6)}$$

where:

$$\overline{v} = \frac{v_{k-1} + v_k}{2} \qquad \text{(VI -7)}$$
$$\overline{\omega} = \frac{\omega_{k-1} + \omega_k}{2} \quad .$$

and the total displacement over the time window *k-u* is:

$$\mathbf{s}_r^u = \mathbf{s}_r^{u \to k} = \sum_{i=0}^{k-u-1} [\mathbf{s}_r^{(k-i-1) \to (k-i)} \mid u] \quad . \qquad \text{(VI -8)}$$

Equation (VI -8) then provides the robot displacement over the time period. The vector **q** is given by the human detector as discussed in Section 7.1. Another transformation needed is to derive the robot position, **p**, relative to the human coordinate frame from the vector **q**. The robot position vector **p** in the human frame can be found through:

$$\mathbf{p}^k = \mathrm{T}_\beta(-\mathbf{q}^k) \quad , \qquad \text{(VI -9)}$$

where β is the angle difference between the human and robot coordinates, as shown in Figure VI-. Because the orientation of the human can't be detected, we assume that the human always faces the direction of his or her displacement. Therefore β at time index *k*, is:

$$\beta = \angle(\mathbf{s}_h^u \mid k) \quad . \qquad \text{(VI -10)}$$

The social positioning behaviour produces the desired robot displacement in the human reference frame. The controlling of the robot is however based on the robot system. The inputs to the motor controller of the robot are the translational speed v and the rotational speed ω. The desired translational speed v_d and the desired rotational speed ω_d of the robot thus are:

$$\begin{aligned} v_d^{k+1} &= c_v f \left\| \mathbf{s}_d^k \right\| \\ \omega_d^{k+1} &= c_\omega f \angle \mathbf{s}_d^k \end{aligned} \quad , \qquad \text{(VI -11)}$$

where f is the system frequency, c_v and c_ω are scaling factors, and $\mathbf{s}_d^k$ is the behaviour output vector measured in the robot frame. That is:

$$\mathbf{s}_d^k = \mathrm{T}_{-\beta} \times \tilde{\mu}(\mathbf{p}^k) \quad , \qquad \text{(VI -12)}$$

where $\tilde{\mu}(\mathbf{p}^k)$ is the social positioning behaviour output, the desired robot displacement in the next time index measured in the human reference frame.

The onboard motor controller is at its most reliable at low and medium speeds. Large errors can be introduced when the control signal v_d and ω_d are too high. Therefore the value of both variables will be scaled down by the scaling factors to ensure robust control of the robot. The value of rotational scalar c_ω is to ensure the rotational velocity is no larger than $\frac{\pi}{4}$ rad/s^{-1}. This practical limitation was discovered by experimental tests. Because $\omega_d \in [-\pi, \pi]$, $c_\omega = \frac{1}{4}$.

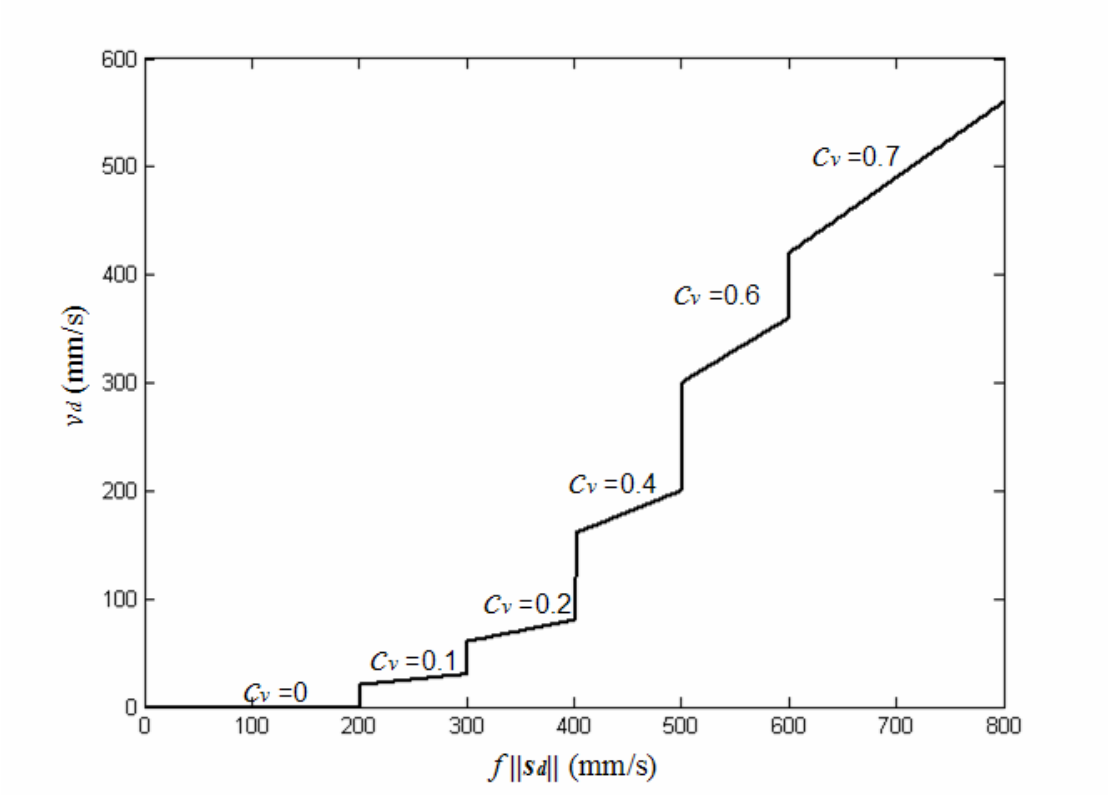


Figure VI -3 The translational speed control profile of the robot. It assigns the robot speed based on the desired robot speed calculated from the behaviour output.

The value of the translational scalar is based on profiling. Figure VI - shows the translational speed profile. Our tests showed that within the limit of 500 mm/s the system is generally robust. But because rotational speed and translational speed are not independent, we also found that the system controls the rotational speed better when the translational speed is low. Therefore the translational speed has been scaled down even when it is lower than 500 mm/s. This speed profile scales down the translational speed heavily when the desired speed is small in order to ensure the steering of the robot is correct near the reward peak because a small desired speed is normally expected near the reward peak. Generally speaking, scaling the speed down below the desired target introduces a time delay for the robot movements, for the sake of robustness. The experiments in Section 7.5 show that this trade-off is acceptable and worthwhile.