Title Framework of Active Robot Learning

Name Beisheng Liu

FRAMEWORK OF ACTIVE ROBOT LEARNING

By

Beisheng Liu

A thesis submitted to the University of Bedfordshire, in fulfilment of the requirements for the degree of Master of Science by research

October, 2008

ABSTRACT

In recent years, cognitive robots have become an attractive research area of Artificial Intelligent (AI). High-order beliefs for cognitive robots regard the robots' thought about their users' intention and preference. The existing approaches to the development of such beliefs through machine learning rely on particular social cues or specifically defined award functions. Therefore, their applications can be limited.

This study carried out primary research on active robot learning (ARL) which facilitates a robot to develop high-order beliefs by actively collecting/discovering evidence it needs. The emphasis is on *active learning*, but not teaching. Hence, social cues and award functions are not necessary. In this study, the framework of ARL was developed. Fuzzy logic was employed in the framework for controlling robot and for identifying high-order beliefs. A simulation environment was set up where a human and a cognitive robot were modelled using MATLAB, and ARL was implemented through simulation.

Simulations were also performed in this study where the human and the robot tried to jointly lift a stick and keep the stick level. The simulation results show that under the framework a robot is able to discover the evidence it needs to confirm its user's intention.

Keywords: cognitive robot, high-order beliefs, robot active learning, fuzzy logic control, MATLAB modelling

DECLARATION

I declare that this thesis is my own unaided work. It is being submitted for the degree of Master of Science by Research at the University of Bedfordshire.

It has not been submitted before for any degree or examination in any other university.

Name of candidate: BEISHENG LIU

Signature: _____

Date: _____

ACKNOWLEDGMENT

I wish to thank to my supervisor: Dr Dayou Li of the Department of Computing and Information Systems, University of Bedfordshire, for his guidance and supervision of the research work and this dissertation.

I also wish to express my thanks to my parents, Mr Huxing Liu and Mrs Li Zhang for their support and encouragement during the year of study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

1.1 Motivation

Robots are automatic devices and are widely used in various areas since 1959 when American scientists developed the first robot in the world. They can be divided into three generations according to automatic control methods used. The first generation of robots do not have sensing devices and uses sequence control method. A user can teach the robot how to complete a task by programming. Those robots are designed to fulfil the heavily repeat tasks. The second generation of robots are equipped with different kinds of sensing devices such as position sensors, velocity sensors, force sensors, etc. The robots are controlled by computers and can perform more complex tasks. The second generation robots also have the ability of self-adjusting and self-adaptive capacity. The third generation of robots have some more complex sensory devices which make the robots able to recognise objects in surrounding environments. The third generation robots also have autonomous decision-making ability and self-learning ability which can make the robot "intelligent".

Cognitive robots are the third generation robots. They are usually used in rehabilitation, home care, therapy, rescue, inspection, maintenance and construction. They will need to autonomously co-work with humans in a sensible and adaptable manner which requires the robot able to recognize their users' intentions and preferences. This means that these robots are expected to possess cognitive capabilities such as knowledge, believes, preference and motivational attitudes.

Robot learning plays an important role in knowledge acquisition, motivation establishment and preference identification. The current robot learning approaches include the imitation learning and reinforcement learning. The imitation based learning uses social cues such as pointing and gazing to indicate what the user intended to do next (Dillmann 2004, Breazeal et al. 2005, Calinon and Billard 2006). The user first teaches a robot by demonstrating gestures, for example, pointing to and gazing an object, to the robot. These gestures serve as social cues of his interest on the object. Then the robot imitates the gestures for the user's approval. This imitation process enables the robot to recognise the user's intention when it captures the same gestures. This imitation based approach has two limitations. First, it only allows the robot to learn the user's intention passively. Second, the users must give exactly the same gestures as they act at the teaching stage to make sure the robot could pick up their intentions.

Tapus and Mataric (2007) proposed a reinforcement learning process for medical care robots. This approach uses the introversion-extroversion level to consider the patients' preference and employs an award function. The award function is defined over the robot's behaviour space. When local optima of this function are reached the robot will be awarded. The award function has become the key to the success of this approach. The use of the award function, on the other hand, limits the application of this approach because the definition depends on how a robot's behaviour is parameterised. For different tasks, this function may have to be defined differently.

An approach which does not rely on social cues and not require specifically defined award functions is needed for robots to develop their cognitive capabilities.

Discovering learning in education, also known as "learn by doing", encourages learners to discover knowledge through performing supervised experiments by themselves. Inspired by the discovering learning, this study proposed an approach of active robot learning (ARL). The proposed ARL allows a robot to perform test actions to its user and to identify the user's intention/preference by analysing her/his responses to the test actions. ARL does not rely on social cues and on explicitly defined award functions. ARL also is an active learning approach. This means that the robot decides when to learn and what to learn.

1.2 Aim and Objectives

This study aims at the development of the framework for ARL, including the components and the relationship between the components. This study also sets up a stick-lifting scenario to test the framework.

Objectives are in the following:

- To investigate the existing methods of the development of cognition capability for robots and their applications through a literature review
- To decide what functions/components are required by ARL
- To decide the relationship between the components
- To develop mathematical models for robot and human in order to test the ARL framework
- To develop a local fuzzy control algorithm for the robot model to perform the task of object lifting

- To implement the prototype of the ARL framework
- To integrate the prototype with the robot and human models as well as fuzzy control algorithms
- To test the prototype in a simple stick-lifting scenario.

1.3 Structure of Dissertation

Chapter 2 provides the survey of state-of-the-art approaches in the area of cognitive robots. This chapter describes four main research areas of cognitive robot and investigates the major approaches in these areas. The four areas are: the structure of cognitive robot, how to let robot comprehend user's intention, the task planning and the environment recognition.

Chapter 3 gives the mathematical model of an industrial robot arm which was used in this study for the purpose of simulation. Manutec r3 industrial robot arm was used as a benchmark to build the dynamic model of the robot arm used in simulation. The robot arm has four links and they are connected by rotational joints. Each link is driven by an electric motor and a gearbox consisting of steel gear wheels embedded in the link. The position and rate of each motor are measured by a tachometer on the motor's axis. The angle between two links can be calculated from the motor position and gear ratio of the corresponding gearbox. Chapter 3 describes the development of the dynamic model of Manutec r3 using MATLAB and SIMULINK, and the setting of parameters.

Chapter 4 presents details of fuzzy logic control system (FLCS) development. The FLCS is used to control the robot arm model (described in Chapter 3) to cooperate with a human, which was modelled by a second robot arm that act according to a predefined trajectory, in lifting a stick and keeping the stick level. The FLC consists of two fuzzy logic controllers. One is to control the robot arm to cooperate with the human model. The other is designed for the purpose of keeping the stick level after the human model stops moving. A switcher is used to switch the two controllers.

Chapter 5 introduces the framework of ARL, including concepts, simulation results and implementation. The framework consists of five components, namely an action bank, an interface engine, a moment determination mechanism, an intention identification mechanism and an intention model. The action bank stores test actions that can be taken to test its cooperative partner. The inference engine reasons about what actions to be taken for a specific purpose. The moment determination mechanism decides the moment of test. The intention identification mechanism interprets responses of the users and identifies intention and preference. The intention model represents intentions to support the intention identification. Testing results are also given in this chapter.

Chapter 6 gives conclusions and further work.

CHAPTER 2. LITERATURE REVIEW

2.1 The Structure of Cognitive Robot

2.1.1 Three-layer structure

Most of the cognitive robotic system comprises functions of perception, memorising and learning, problem-solving (task planning), motor control and communication. The very first cognitive robots can only be able to perform task planning based on sensor readings. The problems these cognitive robots suffer are the long response time and the poor expression of environment. Albus (2000) (also see Burghart et al. (2005)) tried to solve these problems by introducing a three-layer structure cognition system for cognitive robots, as shown in Figure 2.1.

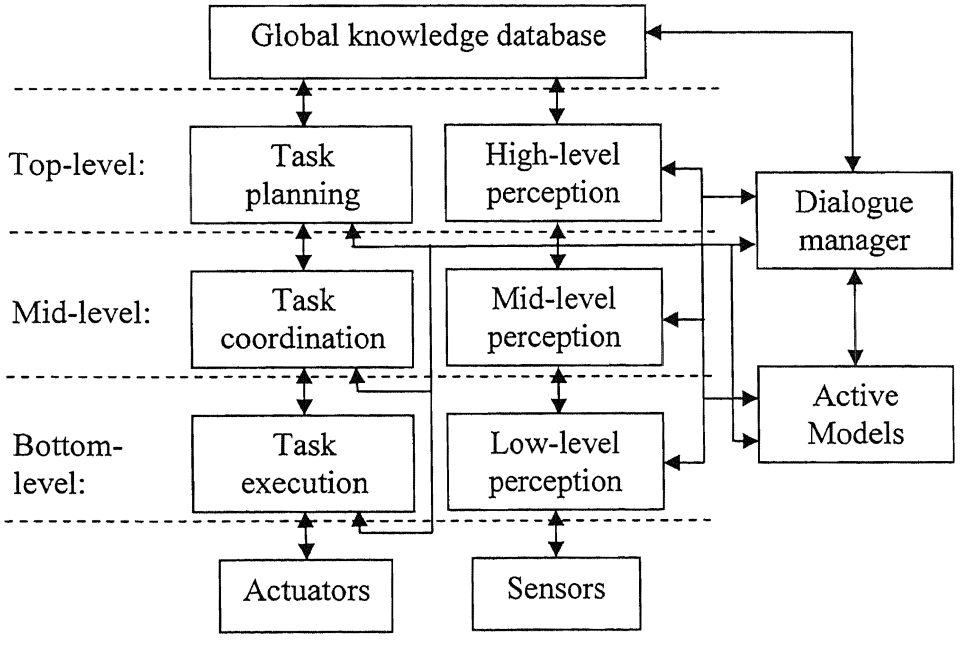

Figure 2.1 Three-layer structure of cognitive robot

The bottom layer consists of a low-level perception module and a task execution module for fast responding. The low-level perception collects sensor readings. Those sensor readings that are relevant to low-level control of the robot will be passed to the task execution module for giving a fast response to the environmental changes. The rest will be sent to the middle layer.

The middle layer consists of a mid-level perception module and a task coordination module. The mid-level perception module comprises various recognition components. These components have the function of multimodal recognition such as audio-visual speaker tracking and have access to the database for using background knowledge stored in the database. The task coordination module receives a sequence of actions which are planned at the top layer. It coordinates the running of all tasks and sends the final correctly parameterized and deadlock free flow of actions to the task execution module at the bottom layer.

The top layer comprises a high-level perception module and a task planning module. The high-level perception module contains all understanding components such as single modality understanding, multimodal fusion, and situation recognition. It interprets actions by the user and creates a situational representation and interpretation for having a high-quality expression of the environment. The task planning module operates in a real-time manner using task knowledge stored in the database. The planning process starts when a desired task has been successfully interpreted out of the data passed from the high-level perception module. A plan consists of a sequence of actions which the task planner selects from a knowledge base. The task planning

module assembles the plan for the intended task and adapts the free parameters to the given task.

In addition to the modules included in the three layers, the structure proposed by Burghart et al. also consists of active models, a dialogue manager, and a global knowledge database. The active models serve as a short time memory and provide current environmental information, as well as information about objects in the focus of attention. The dialogue manager communicates with the user and interpretation of communicative events. It can be initiated by the system to request information by the user. The global knowledge database contains object models, environment model, task knowledge, gesture library, person library, dialogue library, sound library and Hidden Markov Model (HMM).

2.1.2 Eyemind structure

Some cognitive robot structures are based on the "sense-think-action" such as the three-layer structure model described in the previous sub-section. Some others are based on "behaviour". Behaviour based approaches have been successfully applied to dynamic environments. The character of the behaviour based robots is that the robots are self-motivated, that is, the robots desire to explore environments.

Maes (1994) (also see Petitt and Braunel (2003)) introduced a behaviour based cognitive robot structure called Eyemind. It divided a robot's mind model into three classes, namely, Id, Ego and Super-ego, as shown in Figure 2.2.

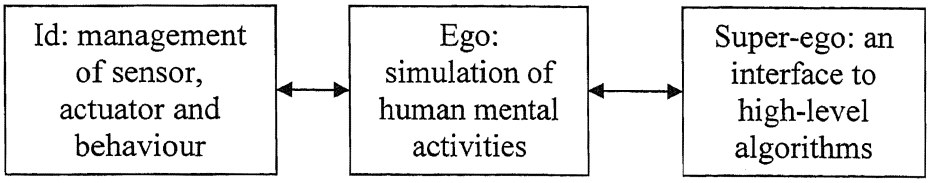

Figure 2.2 Mind model of Eyemind structure

The Id provides the functionality required for managing all the sensors and actuators. It can access actuators and sensors. Based on this accessibility, the Id manages the current behaviours of a robot. Behaviour refers to a mapping from a sensor input to a motor output. The Id allows a robot to combine simple behaviours to assume more complex behaviours. For example, a robot that is able to perform light-beams-following can be deployed to maze solving and navigation in dynamic environments.

In the Id, behaviour can be suppressed or excited by a feedback loop between their sensors and actuators. When the behaviour is excited, the excitation value generated is added to the current excitation value, which is itself the result of previous excitation events. If the new excitation value is greater than a threshold, then the behaviour is activated. The activation of behaviour can take many forms, from creating an output signal for an actuator, requesting sensor input, to triggering other behaviours.

The Id retains a list of up to 16 'root' behaviours. Each of these behaviours is excited by the timer processor unit (TPU) at set intervals. The TPU interrupts the CPU and causes the CPU to execute the list of root behaviours. The root behaviours then either do nothing, or execute their specific fire function.

The Ego simulates human's mental activities. For example, when someone wants to open a door, he will decide to push or rotate the handle in a very short period of time. If the door didn't open, he will do it in another way. The Ego module acts similarly to this logic which is illustrated in the following pseudo-code:

```
while (desired_states)
{
  for each state
  {
    if Criticise (past_states, current_states, desired_states)
    {
      LearnBad (superego);
      RemoveState (state);
    }
    else if Satisfied (id)
    {
      LearnGood (superego);
      RemoveState (state);
    }
    else state->Satisfy ();
  }
}
Superego->CreateStrategy ();
```

Figure 2.3 Pseudo-code of the Ego

The key in the Ego is the Criticise ( ) function. This function tests whether the current state reaches the desired state in the specified period of time. When the desired state is

reached, the robot will continue to do the task. If the current time exceed the desired time of task finished, the system will send back an error signal and reform the strategy.

The Super-ego houses no real information, but provides an interface with higher-level algorithms, such as expert system and adaptive critics. When the CreateStrategy ( ) function is called, a list of states which correspond to the strategy is appended to the list of desired states.

2.1.3 iCub cognitive system structure

Sandini et al. (2006) developed a cognitive structure for iCub, an open platform for robot simulation. The structure has three parts, namely, a network of perceptuo-motor circuits, a modulation circuit which affects homeostatic actions selection by disinhibiting the perceptuo-motor circuits, and a system to affect anticipation through perception-action simulation.

The anticipatory system allows a cognitive robot to rehearse hypothetical scenarios and in turn to influence the modulation of the network of perceptuo-motor circuits. Each perceptuo-motor circuit has its own limited representational framework and together they constitute the phylogenetic abilities of the system. The modulation circuit carries out self-modification in terms of parameter adjustment of the phylogenetic skills through learning and developmental adjustment of the structure and the organization of the robot. This enables the cognitive robot to alter its own dynamics based on experience, to expand its repertoire of actions, and thereby adapt itself to new circumstances.

The iCub cognitive architecture is illustrated in Figure 2.4.

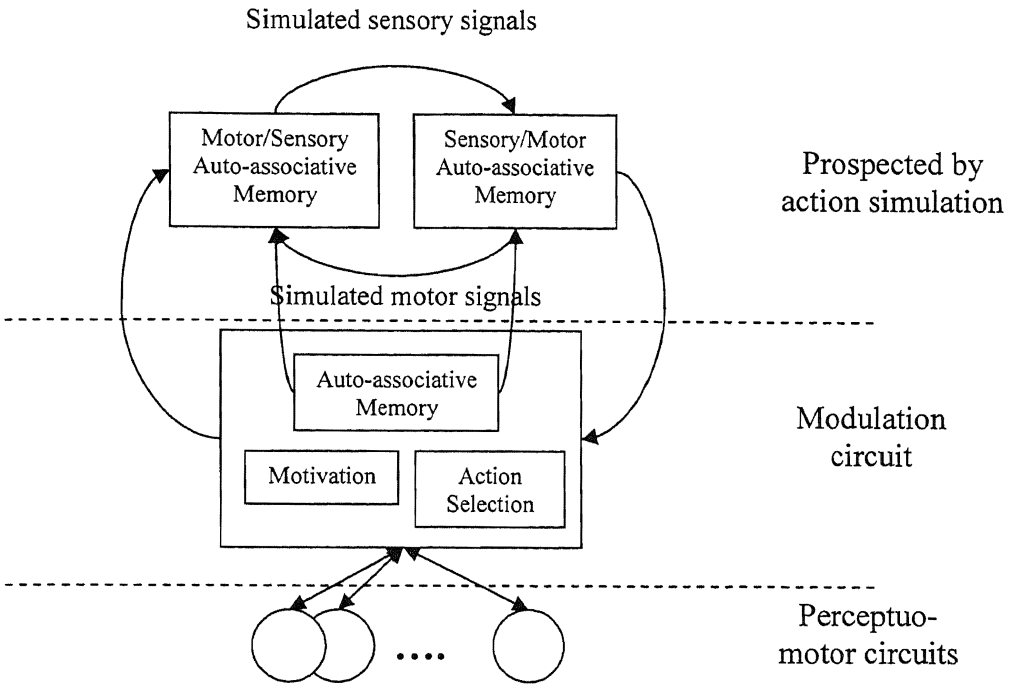

Figure 2.4 iCub cognitive architecture

2.1.4 Behaviour based hierarchical structure

Arkin (1998) interpreted behaviour as a pair of attention and intention. Attention prioritises tasks and provides some organization in the use of sensorial resources. Intention, on the other hand, determines the behaviours to be activated.

Based on Arkin's interpretation, Duro et al. (2003) developed a behaviour based hierarchical structure for cognitive robots. In this structure, behaviours are classified into two categories: lower-level behaviours and higher-level ones (also known as complex behaviours). This structure uses the concept of attention to prioritise a task and then form a higher-level controller. This controller, based on the concept of

intention, is able to choose lower-level behaviours to form a complex behaviour. This process can be described as following: A designer must provides the robot with whatever behaviours he or she decides that may be useful. This initial behaviour set may not be complete and may include unnecessary behaviours. The high-level controller uses the data from sensors, which reflect the state of the environment, and other controllers to choose behaviours.

To prevent the problem of the designers having to determine all the necessary lower level behaviours, this approach includes the possibility of cooperatively coevolving lower and higher level behaviours. That is, a higher-level behaviour may be evolved by itself using previously evolved lower level behaviours, or it may be coevolved with part of the lower level behaviours and use the previously evolved ones. When the designer is faced with a problem where he is only able to identify part of the behaviours that may be involved, the unidentified ones will be evolved at the same time as the higher-level controller. This is illustrated in Figure 2.5.

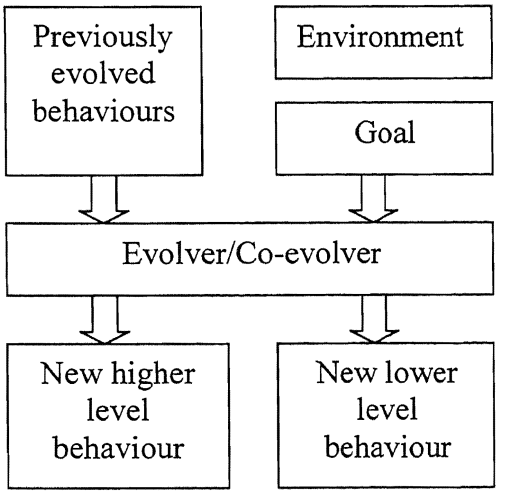

Figure 2.5 New behaviours evolution

2.2 User Intention Cognition through Learning

2.2.1 Learn by imitation

Robot learning plays an important role in background knowledge building, motivation establishment and preference identification. The current robot learning approaches include imitation learning. The imitation based learning uses social cues such as pointing and gazing to indicate what the user intended to do next (Dillmann 2004, Breazeal et al. 2005, Calinon and Billard 2006). The user first teaches a robot by demonstrating gestures, for example, pointing to and gazing an object, to the robot. These gestures serve as social cues of his interest on the object. Then the robot imitates the gestures for the user's approval. This imitation process enables the robot to recognise the user's intention when it captures the same gestures.

Experiments carried out in Calinon and Billard (2006) can be described as below: During a first phase of the interaction, the designer demonstrated a gesture in front of a robot. The robot then observed the designer's gesture. Joint angles trajectories are collected from a motion sensor. The second phase was begun when the robot collected the different movements of user. The robot compared the gesture it collected with the gesture stored earlier and finds the cues of them. Then the robot pointed at an object that the user most likely to be interested. The robot then turned to user for evolution of its selection. The designer signals to the robot whether the same object has been selected by nodding/shaking his/her head.

In the imitation learning, a Hidden Markov Model (HMM) with full covariance matrix is used to extract the characteristics of different gestures which are used later to recognise gestures from the user. The characteristic of a gesture is expressed by transition across the state of the HMM. Using such a model requires the estimation of a large set of parameters. An Expectation-Maximisation (EM) algorithm is used to estimate the HMM parameters. The estimation starts from initial estimates and converges to a local maximum of a likelihood function. It first performs a rough clustering. Next, EM is carried out to estimate a Gaussian Mixture Model (GMM). Finally, the transitions across the states are encoded in a HMM created with the GMM state distribution.

2.2.2 Learn by conversation

The most direct way to let the robot to understand the users' intention is conversation. Hassch et al. (2004) developed a Bielefeld Robot Companion (BIRON) which is a robot who accompanies to a human. It consists of cameras, microphones, laser range finder, speech recognition system, and other components. This robot is able to understand its users' intention through oral instructions and observation of the user's sight.

BIRON employs a human concern system to decide which user is interested by the robot. When someone is talking while watching the robot, the robot's attention will be transferred to this people. When individuals are talking at the same time and no one is watching the robot, the robot will pay attention to the people who has not been concerned for the longest time.

A Dialogue Manager is also included in the robot, which is responsible for receiving the instructions from users. The Dialogue Manager could interact with users and solve some ambiguous question by asking them.

A speech recognition system is used to understand users' intention by analysing received sound information from the microphone. The two major challenges of the speech recognition system are:

- The speech recognition has to be performed on distant speech data recorded by two microphones
- Speech recognition has to deal with spontaneous speech phenomena.

The recognition of distant speech with two microphones is achieved by reconstructing a single channel representation of the speech originating from a known location on the basis of the different channels recorded by the microphones (Leese 2002).

The speech understanding components deals with spontaneous speech phenomena in conversations between a user and the robot. For example, large pauses and incomplete utterances can occur in such task oriented and embodied communication. However, missing information in an utterance can often be acquired from the scene. For example the utterance "Look at this" and pointing gestures to the table concludes to the meaning "Look at the table".

2.2.3 Reinforcement learning system

Tapus and Mataric (2007) proposed a reinforcement learning based approach to robot behaviour adaptation. The aim of this approach is to develop a robotic system capable of adapting its behaviours according to the user's personality, preference, and profile in order to provide an engaging and motivating customised protocol.

In this learning approach, a robot incrementally adapts its behaviour and its expressed personality as a function of the user's extroversion-introversion level and the amount of performed exercises. Then the robot attempt to maximize that function.

The learning process consists of the following steps:

- Parameterisation of the behaviour
- Approximation of the gradient of the reward function in the parameter space
- Movement towards local optimum.

The main goal of this robot behaviour adaptation system is to optimise three main parameters (interaction distance, speed, and verbal cues) that define the behaviour of a robot, so that the robot can adapt itself to the user's personality and improve its task performance. Task performance is measured as the number of exercises performed in a given period of time. The learning system changes the robot's personality which is expressed through the robot's behaviour to maximise the task performance.

2.3 Task Planning of Cognitive Robot

2.3.1 Forward model

Hashimoto et al. (1992) (also see Dearden and Demiris (2005)) developed a system that enables a robot to autonomously learn a forward model with no prior knowledge about its motor system and about external environment. Information about the effects of the robot's actions is captured by a vision system. The vision system generates a cluster of image features and the robot will automatically find and track moving objects in a scene.

At the beginning, the robot generates random motor commands to its motor system and receives information back from the vision system. The information is used to learn the structure and parameters of a Bayesian network which represents the forward model. This model can then be used to enable the robot to predict the effects of its actions. The robot learning process is shown below:

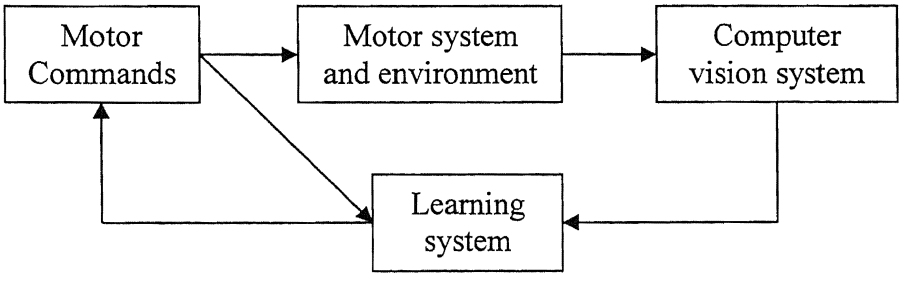

Figure 2.6 Robot learning process

The Bayesian network is an ideal way to represent forward model. The foundation of the Bayesian network is based on the state of a robot, the robot's motor commands

and the observations of the robot's states that are received from the vision system. The learning system of the robot is aimed at learning the casual associations between them. The structure of the Bayesian network is shown in Figure 2.7. The question marks in the diagram represent parameters to be determined through the learning process.

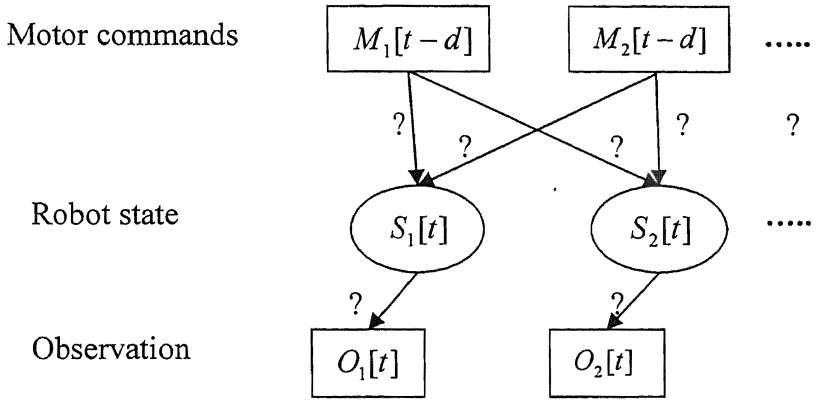

Figure 2.7 Bayesian network structure

No prior information about what to track is available at the beginning of the learning process as the environment is unknown. The robot needs to find and track the moving objects by capturing their positions and velocities using its vision system. After the observations are received, the robot can then realise how its motor commands interact with the state and adjusts the parameters of the Bayesian network using the difference between the actual state and the desired state.

2.3.2 Inverse dynamics robot trajectory learning

The approach of inverse dynamics robot trajectory learning was developed by Robbel and Vijayakumar (2007). The key to the success of this approach is the generation of

a smooth trajectory for robot arms with different degrees of freedom. Most of the current studies are based on the storage of multiple trajectories and employ a training system to choose one of the trajectories and to amend the parameters to achieve a given goal.

In order to get a highly effective inverse dynamic robot arm trajectory exploration strategy, Robbel and Vijayakumar developed the robot learning system. It consists of a feed-forward model of the inverse dynamics and a corrective PID controller. The difficulties with data selection for robot control are in twofold. First, points cannot be chosen freely from the input distribution. Second, the inverse dynamics of a system cannot be learnt easily online.

The learning process of this system can be described as below:

1. At every time step, model prediction and prediction confidence for the current query point x_q are manually determined. The model generalization error is also postulated manually by the size of the confidence intervals.
2. If the confidence is above a threshold, the model prediction is applied as a control signal to all joints and continues with step 1. Otherwise, set the last trusted point x_{q-1}, which the model predicts, as a set point.
3. Execute a number of directed exploratory actions around x_{q-1} to reduce the confidence interval size. Those actions are followed by resetting the arm to the set point via PID control. Then continue with step 1.

2.4 Environment Recognition

2.4.1 Map building

Robots are expected to work at humans' home in the future. This requires the robots to develop abilities to understand, interpret and represent environments where they are deployed. Vasudevan et al. (2006) developed a probabilistic approach which is able to represent in-door environments. This approach is based on the location of objects and the relationship between objects. A global presentation consists of a number of local representations which represent places. Objects in a place are detected and used to build a local map (representation) in the form of a local probabilistic object graph. Doors are identified and used as links to connect the local ones to form the global representation.

The process of building up a global map (representation) can be shown in Figure 2.8. The process begins with local representation development, including object detection, recognition and probabilistic object graph development. When a local map is built up, the process starts to extract doors (also known as high-level features). It then moves to a new place and develop a new local map which is connected to the doors identified.

This representation must consider and handle uncertainties existing in the perception of a robot. For this reason, the representation is probabilistic. "Existential" beliefs are obtained for each object that is observed. Simultaneously, precision beliefs are maintained in the form of covariance matrices. These beliefs are based on detailed mathematical formulations given below:

$$X_0 = f(X_c) \qquad (2.1)$$

where $f = M_{RO} \times M_{CR}$, M_{RO} stands for transformation between robot frame and absolute reference, and M_{CR} stands for transformation between camera frame and robot frame.

A precision belief is presented in the form of a covariance:

$$P_0 = F_1 P_1 F_1^{'} + F_2 P_2 F_2^{'} \qquad (2.2)$$

where $F_1 = J_{X_1}(f)$, $F_2 = J_{X_2}(f)$, $X_1 = (X_R, Y_R, \theta_R)$ represents robot pose, $X_2 = (X_C, Y_C, \theta_C)$ stands for object position in camera frame, P_1 is the covariance matrix which represents uncertainty in robot position, P_2 is the covariance matrix which represents uncertainty in the object position, F_1 is the Jacobian of "f" with respect to X_1, and F_2 is the Jacobian of "f" with respect to X_2.

The belief representation for relationships between objects is shown below:

Let $(X_1, Y_1, Z_1) and (X_2, Y_2, Z_2)$ represent two objects, $f(X_1, X_2)$ stands for relative spatial information between the two objects, P_1 and P_2 stands for uncertainty in object positions (covariance matrices).

The precision belief is defined as:

$$Bel_1(f) = F_1 P_1 F_1^{'} + F_2 P_2 F_2^{'} \qquad (2.3)$$

where $F_1 = J_{X_1}(f) \,\&\, F_2 = J_{X_2}(f)$ are the Jacobian of "f" with respect to X_1 and X_2 respectively and existential belief $Bel_2(f) = \min$ (belief in existence of objects).

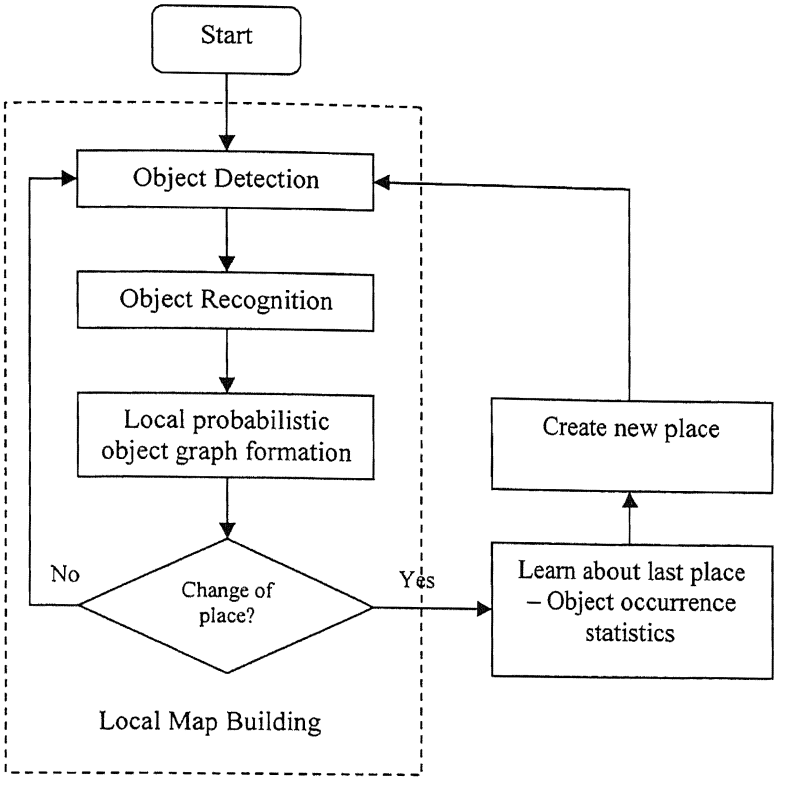

Figure 2.8 Map building process

Figure 2.9 shows the place cognition process using the created maps, where C represents "Change" and R represents "Recognition". The first step of reasoning process is place classification. The robot uses the object it perceives to classify the place into one of its known place categories (office, kitchen etc.). Next step is recognizing specific instants of the place it is aware of (place recognition). Accordingly map update or adding of new place is done.

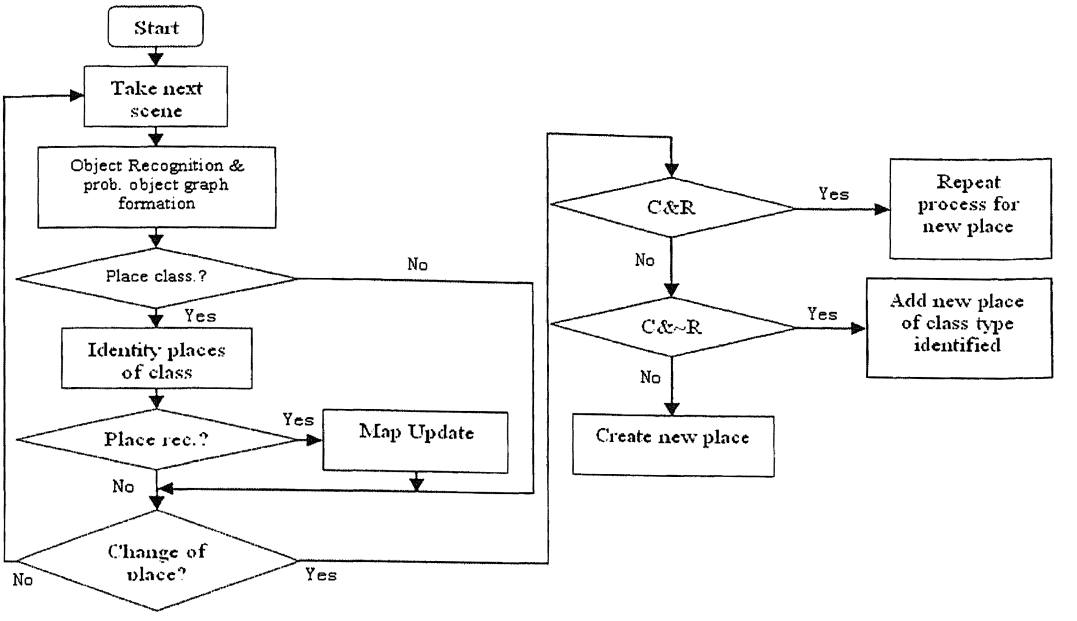

Figure 2.9 Place recognition process

2.4.2 Dynamic environment modelling

Benjamin et al. (2007) developed a cognitive structure called Adaptive Dynamics and Active Perception for Thought (ADAPT). ADAPT uses Ogre3D, an open source gaming platform, and Soar, a problem-solving tool which is able to perform symbolic reasoning, to develop the model of a dynamic environment, called world model. The ADAPT's environment modelling system enables a robot to model its environment. The structure of this system is illustrated in Figure 2.10.

ADAPT's world model is a graphic view of the environment. It is saved in Orge. This world model is not directly connected to the real world. Instead, it is linked to Soar which has a connection to cameras, as can be seen from Figure 2.10.

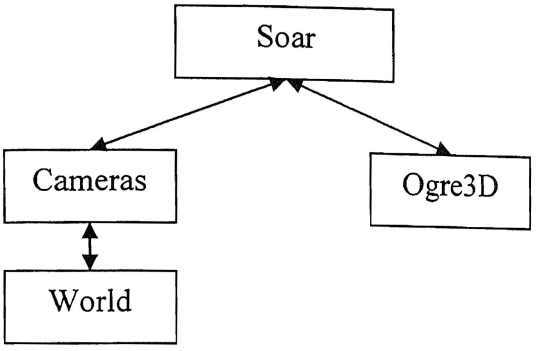

Figure 2.10 Environment modelling system

Orge embodies the graphical and dynamic aspects of the world model and Soar contains the symbolic part of the world model. The way in which the system works can be explained using the following example. When Soar recognises a person sitting in a chair (image captured by cameras), it will construct virtual copies of the chair and the person in Orge and create symbolic structures in Soar's working memory pointing to them, as well as a symbol structure for the relationship of sitting. Orge serves as the model that interprets the symbols in Soar's working memory. The relationships between the Soar and Orge parts of the world model are updated automatically each Soar cycle.

The dynamics is modelled in the way that Soar continuously tests for significant differences between the expected view and the actual view. After the graphic of the environment in Orge is segmented and placed into Soar's working memory, Soar starts to operate such tests. If a new object appears, Soar will propose a new operation to a robot to look at this object and try to recognise it. The robot will then turn its cameras towards this object and then call its recognition software to process a visual field that contains the object.

Once the object is recognised, a virtual copy is created in Orge. If the object from the visual field approximately match one of the expected objects from Orge, ADAPT assumes it is the same object. Otherwise, the object will be added to the world model.

CHAPTER 3. THE DYNAMIC MODEL OF THE ROBOT ARM

3.1 Siemens Manutec r3 Robot Arm

Siemens Manutec r3 is an industrial robot arm. It is often used for computer aided control system design and dynamic trajectory planning. The robot arm is shown in Figure 3.1. Siemens Manutec r3 robot arm can simulate a number of physical effects, such as the robot arm movement, friction, elasticity and damping. The links of the robot arm is driven by motors and the motors are controlled by controllers.

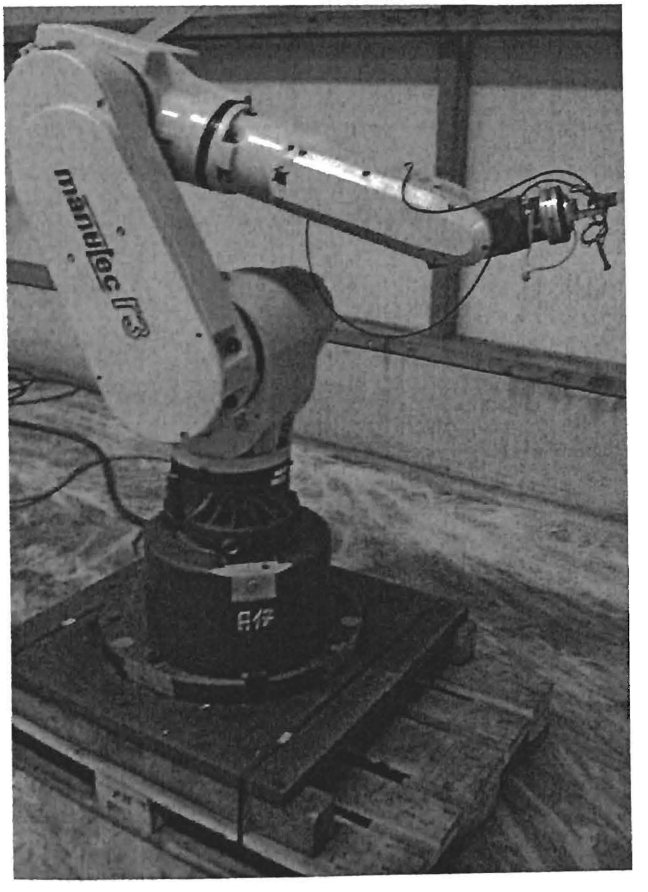

Figure 3.1 Siemens Manutec r3 robot arm

3.1.1 Dynamic model of Siemens Manutec r3 robot arm

The dynamic model of the entire robot arm is shown as below:

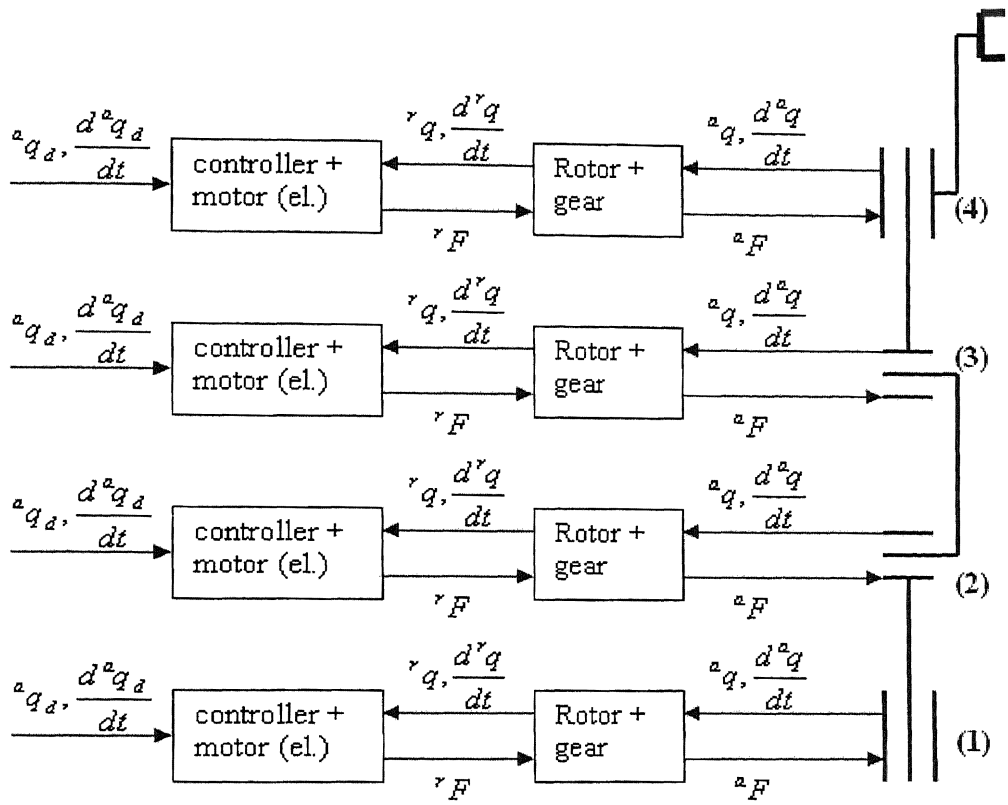

Figure 3.2 Dynamic model of Siemens Manutec r3 robot arm

The meanings of the parameters shown in Figure 3.2 are:

${}^{a}q_{d}$ is the desired angle of revolute joint in [*rad*]

${}^{a}q$ is the angle of revolute joint in [*rad*]

${}^{r}q$ is the angle of rotor in [*rad*]

$d({}^{a}q_{d})/dt$ is the desired angular rate of revolute joint in [*rad/s*]

$d({}^{a}q)/dt$ is the angular rate of revolute joint in [*rad/s*]

$d({}^{r}q)/dt$ is the angular rate of rotor in [*rad/s*]

p is the gear ratio, i.e. if elasticity is neglected in the joint, $q = p \cdot {}^{a}q$

${}^{r}F$ is the torque in air gap of motor in $[Nm]$

${}^{a}F$ is torque in joint in $[Nm]$

The robot arm has four links. They are connected by rotational joints. Each link is driven by electric motor and a gearbox consisting of steel gear wheels embedded in the link. The position and rate of each motor is measured by an encoder amounted on the motor's axis. The angle between two adjacent links is calculated from the motor position and gear ratio of the corresponding gearbox. Thus the position of the end-effector can be measured. Every joint of the robot arm is driven by a torque (${}^{a}F$), which is produced by the corresponding motor and transformed to the joint via a gearbox. To simplify the dynamic model, the rotor of the motor and the gear wheels are treated as one rigid body with rotational symmetry, called rotor. It is assumed that the complete friction is acting at the rotor. Further more, there is no dynamic coupling between the rotors and the links of the robot in the model.

The current of motor is controlled by "controller+motor (el.)". The module of "rotor+gear" contains the mechanical part of motor and gearbox. The internal forces of the motor and the gear are described by rotor. To simplify the model, the elastic deformation of the links can be ignored.

The motors and the controllers inside the joints of the robot arm have the same structure but with different parameters. The motor is an electric rectifier synchronous motor. Because the robot arm uses electric rectification and current control, the dynamics model of motor is the same as a DC motor. The angle and angular rate of

the rotor are measured by an incremental encoder. The module also includes a low pass filter. The entire dynamic model of motor is a three-input structure. The rotational rate measured by a tachometer is sent to the input of rate controller. The angle of the rotor is sent to the position controller as an input. The outputs of the position controller and the rate controller are converted by D/A converter and the sampling interval is 0.008s. The maximum torque of the motor is $|^{r}F|_{max} = 9Nm$. The maximum of continuous working torque is $|^{r}F|_{nom} = 4Nm$. The maximum rotational rate $|^{r}q|$ is about 3000 to 3500 revolutions per minutes (around 315 to 366*rad/s*). The electrical part of the motor and the controller are shown in Figure 3.3.

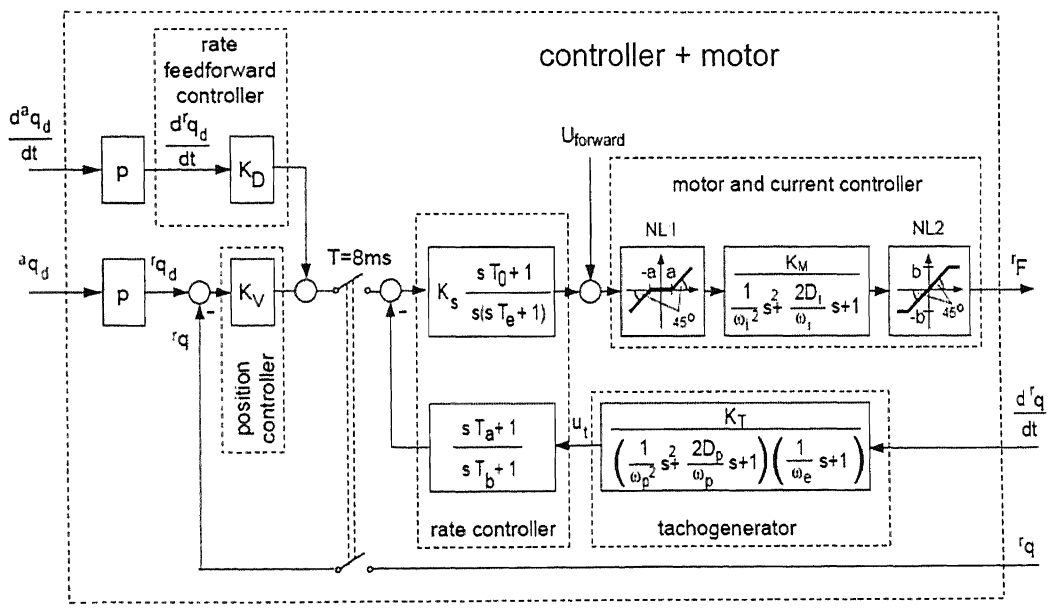

Figure 3.3 Electrical part of the motor and controller

The parameters of the controller and the electrical part of motor are given in Table 3.1:

An actuator drives a rotor to produce torque. The angular velocity of the rotor is transformed to low-speed by gear box and drives the link of robot arm. The rotor,

gearbox and actuator all have friction. The first three links of the robot arm have a certain amount of damping and backlash. In the last actuator, the backlash is comparatively small, such that it can be neglected.

Table 3.1 The parameters of the controller and the electrical part of motor

		unit	Arm1	Arm2	Arm3	Arm4
Position Controller	K_V	-	0.3			
Feed-forward Controller	K_D	-	0.03			
Rate Controller	K_S	-	340.8			
	T_0	s	$9.95*10^{-3}$			
	T_e	s	$0.56*10^{-3}$			
	T_a	s	$40*10^{-3}$			
	T_b	s	$20.2*10^{-3}$			
Tacho generator	K_T	*Vs/rad*	0.03			
	ω_p	*1/s*	2014			
	D_p	-	0.294			
	ω_e	*1/s*	1180			
Motor and Current Controller	K_M	*Nm/V*	1.1616	1.1616	1.1616	0.2365
	ω_i	*1/s*	4590	5500	5500	6250
	D_i	-	0.6	0.6	0.6	0.55
	a	-	0.094	0.094	0.094	0.022
	b	-	9.0			
	$\lvert {}^rF \rvert_{max}$	*Nm*	9.0			
	$\lvert {}^rF \rvert_{nom}$	*Nm*	4.0			
	$\lvert {}^r\dot{q} \rvert_{max}$	*rad/s*	315	315	315	335
Gear ratio	p	-	-105	210	60	-99

To simplify the dynamic model, the motor and the gearbox can be seen as a rotational rigid body, called rotor. Furthermore, it is assumed that the complete friction of the actuator is acting at the rotor. Besides, the coordinates of first three joints' rotation axis are established according to the entire world rather than the former links' end.

There is no dynamic coupling between two links and the centrifugal force and gravity are also neglected so the dynamic model of robot arm is greatly simplified.

The dynamic functions of the Manutec r3 robot arm are given in the following:

$$\begin{vmatrix} {}^{aa}M_{ij} & {}^{ar}M_{ij} \\ {}^{ar}M_{ji} & {}^{rr}M_{ij} \end{vmatrix} \begin{vmatrix} {}^{a}\ddot{q}^{j} \\ \frac{1}{p^{j}} \cdot {}^{r}\ddot{q}^{j} \end{vmatrix} = \begin{vmatrix} {}^{a}h^{i} \\ {}^{r}h^{i} \end{vmatrix} + \begin{vmatrix} {}^{a}F^{i} \\ -{}^{a}F^{i} \end{vmatrix}, \quad i,j = 1,2,3,4 \tag{3.1}$$

$${}^{ar}M_{ij} = \begin{cases} p^{j} \cdot {}^{r}J^{j} \cdot {}^{a}\vec{n}^{i} \otimes {}^{r}\vec{n}^{j} & for\ i < j \\ 0 & for\ i \geq j \end{cases} \tag{3.2}$$

$${}^{rr}M_{ij} = \begin{cases} p^{j} \cdot {}^{r}J^{j} \cdot p^{j} & for\ i = j \\ 0 & for\ i \neq j \end{cases} \tag{3.3}$$

where

${}^{a}\ddot{q}^{j}$ is the angular acceleration of joint j

${}^{r}\ddot{q}^{j}$ is the angular acceleration of rotor j with respect to the link the rotor resides

p^{j} is the gear ratio of the actuator driving joint j

${}^{r}J^{j}$ is the moment of inertia of rotor j with respect to its axis of rotation

${}^{a}\vec{n}^{i}$ is a unit vector which lies in the axis of rotation of joint i

${}^{r}\vec{n}^{i}$ is a unit vector which lies in the axis of rotation of rotor j

${}^{a}F^{i}$ is the applied torque acting in joint i

$\otimes$ is the scalar product of two vectors

${}^{a}h^{i}$ is the friction acting in joint i

${}^{r}h^{i}$ is the friction acting in rotor i.

After neglecting the term $p \cdot {}^{r}J$ for every actuator, the off-diagonal terms ${}^{ar}M_{ji}$'s magnitudes are smaller than the dialogue terms $(p^{j})^{2} \cdot {}^{r}J^{j}$ due to the high gear

ratios p^j. The model equations simplify considerably, because the equations of the robots become decoupled from the rest of the multi body system. Furthermore, the friction torques only appear in the rotor equations and thereby are also decoupled from the rest of the multi body system.

For the last link of the robot arm, since the elasticity in the corresponding gearboxes is neglected in the model the joint and rotor angles are rigidly coupled by the equation below:

$$ {}^{r}q^{j} = p^{j} \cdot {}^{a}q^{j} \qquad (3.4) $$

Figures 3.4 and 3.5 show the structures of the rotors and gearboxes. ${}^{a}L$ in Figure 3.5 represents the coupling torque due to the movement of other joints. The corresponding model parameters are given in Table 3.2. Due to the elasticity, a spring constant c and a damping factor d are presented.

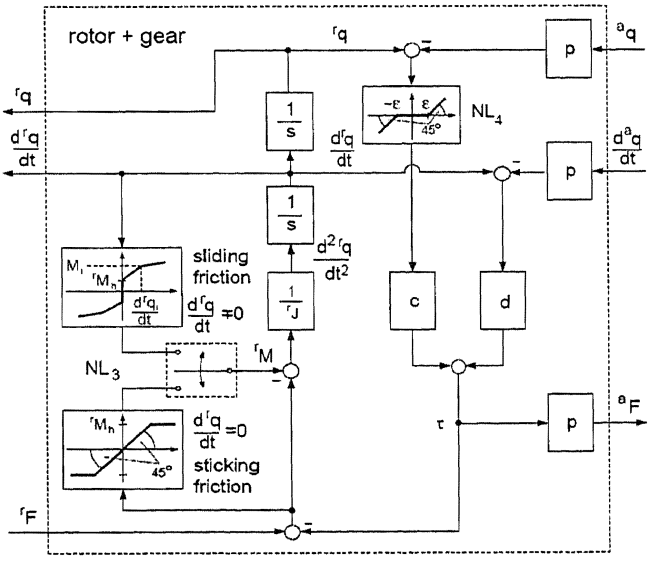

Figure 3.4 The rotor and gear model structure of joint actuators 1, 2, and 3

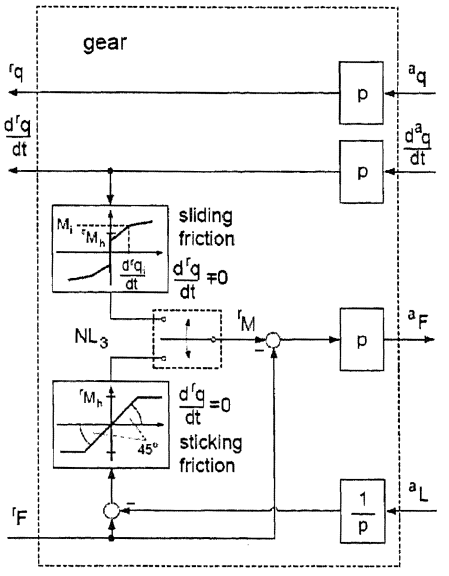

Figure 3.5 The gear model structure of joint actuator 4

Table 3.2 Rotor and gear parameters

	unit	Arm 1	Arm 2	Arm 3	Arm 4
${}^{r}J$	kgm^2	0.0013	0.0013	0.0013	-
ϵ	rad	0.01	0.06	0.0	-
C	Nm/rad	43	8.0	58	-
d	Nms/rad	0.005	0.01	0.04	-
p	-	-105	210	60	-99
${}^{r}M_h$	Nm	0.4	0.5	0.7	0.27
${}^{r}M_1$	Nm	0.4	0.5	0.7	0.22
dq_1/dt	rad/s	0	0	0	0
${}^{r}M_2$	Nm	0.53	0.6	0.9	0.52
dq_2/dt	rad/s	160	130	130	300
${}^{r}M_3$	Nm	-	0.7	-	1.0
dq_3/dt	rad/s	-	360	360	-

3.1.2 Friction model of joint

The friction model has discontinuous and nonlinear characteristic. If the angular rate of the rotor is ${}^{r}\dot{q} \neq 0$, friction acts as an applied torque ${}^{r}M$ according to a nonlinear

function which is approximated by two or three linearly interpolated points. If the angular rate is ${}^r\dot{q} = 0$, two possibilities exist: the friction acts as constraint torque ${}^rM$, for example it compensates the sum of all other torques acting on the rotor ($={}^rF - \tau$) and it forces ${}^r\ddot{q} = 0$, so the friction torque it provided is less than or equal to the upper limit ${}^rM_h$.

If $|{}^rF - \tau| > {}^rM_h$ the friction remains an applied torque but switched to the opposite branch of the nonlinear function. This effect can be explained as follows: assume that ${}^r\dot{q}$ becomes zero from the positive side, the only possibility is ${}^r\ddot{q} \leq 0$, therefore ${}^rF - \tau < {}^rM$. On the other hand, for $|{}^rF - \tau| > {}^rM_h$, ${}^rF - \tau < -{}^rM_h$ is true. This makes ${}^r\ddot{q}$ remains less than zero, and therefore, angular rate ${}^r\dot{q}$ may become negative. The flow-diagram for the computation of friction is shown in Figure 3.6 ($\pm {}^rM_1$ is the value of the sliding friction at zero angular rate).

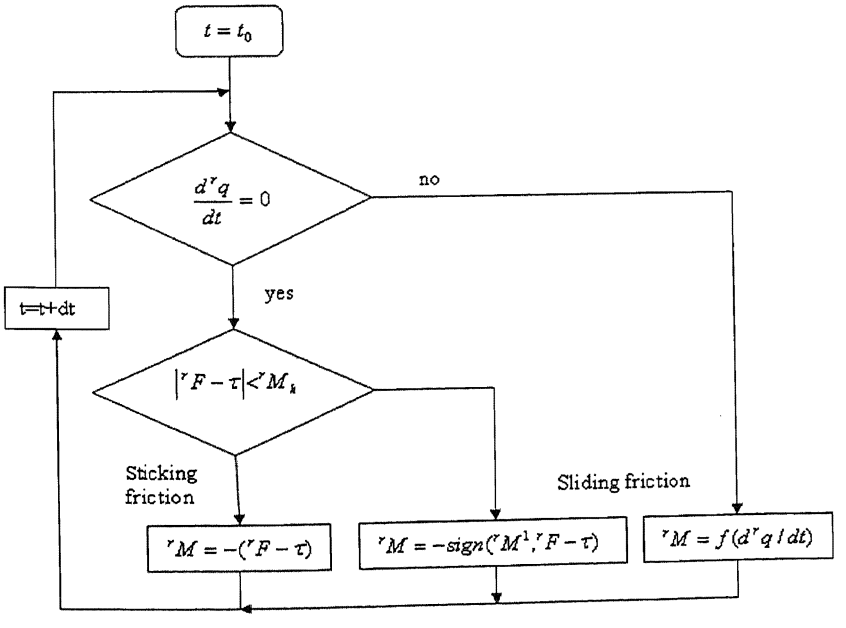

Figure 3.6 Flow-diagram for the computation of friction

In simulation, friction should be handled by state events in the following way: If the angular rate of the rotor ${}^{r}\dot{q} \neq 0$, the angular rate is then used as an indicator function. If this function passes through zero, the crossing time point is determined by the integrator and the integration is stopped to check whether the simulation has continued with sticking or sliding functions. Then the integration restarted. In the case of sticking friction, the function ${}^{r}M_h - \left|{}^{r}F - \tau\right|$ is used as an indicator for the next state event to switch back to the sliding friction model.

3.2 Mathematical Model for Siemens Manutec r3 Robot Arm

3.2.1 Mathematical model development

SIMULINK is a software package that is used for building up mathematic models for dynamic systems and system simulation. It can be used to build a linear/non-linear system or continuous/discontinuous system. The SIMULINK can also be used to build systems with different sampling rates.

In SIMULINK, a mathematical model is represented in the form of block diagram. SIMULINK provides various blocks for modelling a system. Regarding to modelling a robot arm, SIMULINK provides the following blocks:

- Body block: Represents a user-defined rigid body. Body defined by mass, inertia tensor and coordinate origins and axes for centre of gravity and other user specified body coordinate systems

- Inertial frame block: Grounds one side of a joint to a fixed location in the world coordinate system
- Environment block: Defines the mechanical simulation environment for the machine to which the block is connected. The settings include gravity, dimensionality, analysis mode, constraint solver type tolerance, linearization and visualization
- Rotational freedom block: Represents one rotational degree of freedom. The follower body rotates relative to the base body about a single rotational axis going through collocated body coordinate system origins
- Coulomb friction block: Actuates a joint primitive with friction force/torque. Lock if static friction remains within the range of forward and reverse friction limits.

The model of Siemens Manutec r3 robot arm was developed using SIMULINK. The block diagram of this model is given in Figure 3.7.

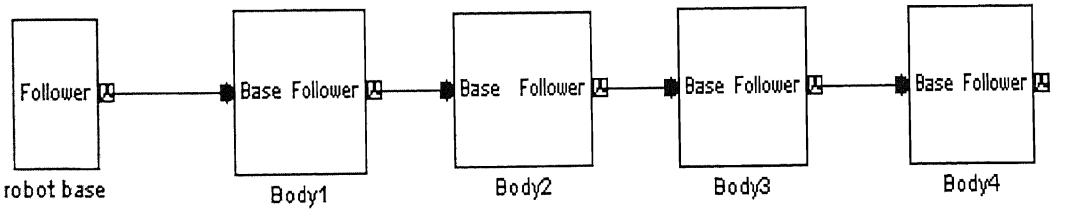

Figure 3.7 Mathematical model of Siemens Manutec r3 robot arm (block diagram)

The robot arm has four bodies and a base. Each body consists of a link and a joint. The bodies and the base are connected together in the way shown in Figure 3.7.

The details of the base are shown in Figure 3.8.

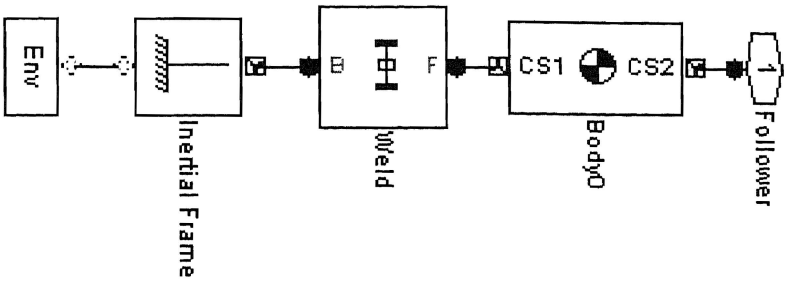

Figure 3.8 Structure of the base

The base consists of an environment block, an inertial frame block, a weld block, a body0 block and a connecting point (Follower). The Body0 block defines the position, weight, and shape information of the base. The base is a $40kg$ homogeneous hexahedral. The settings of the shape parameters of the base are given in Table 3.3.

Table 3.3 Parameters of the base

Show Port	Port Side	Name	Origin Position Vector [x y z]	Units	Translated from Origin of	Components in Axes of
☐	Bottom	CG	[0 -0.2875 0]	m	World	World
☑	Bottom	CS1	[0 -0.4 0]	m	World	World
☑	Top	CS2	[0 -0.1750 0]	m	World	World
☐	Top	CS3	[0.3 -0.4 -0.3]	m	World	World
☐	Top	CS4	[-0.3 -0.4 -0.3]	m	World	World
☐	Bottom	CS9	[-0.3 -0.4 0.3]	m	World	World
☐	Top	CS10	[0.3 -0.4 0.3]	m	World	World
☐	Top	CS5	[-0.3 -0.1750 -0.3]	m	World	World
☐	Top	CS6	[-0.3 -0.1750 0.3]	m	World	World
☐	Top	CS7	[0.3 -0.1750 0.3]	m	World	World
☐	Top	CS8	[0.3 -0.1750 -0.3]	m	World	World

The Origin Position Vector column shows the position parameters of the base and all of the reference points are translated from the origin of world coordinator.

The weld block is used to connect the base and ground. It can be consider as a joint with zero degree of freedom. The inertial frame block defines a fixed point in absolute space. All the movement of objects in this space is referred to this point. The inertial frame block can never be able to move. Sensors cannot be connected to this block.

The environment block defines the settings of environment. These settings include:

- Settings that control how the model is simulated and define the gravity, system dimensionality, analysis mode and tolerance
- Settings that control how constraints are interpreted
- Settings that control how linearization is implemented and define the type and size of perturbation
- Settings that decide whether the machine is displayed in SIMULINK visualization.

Any of the bodies consists of a body block, a driver module, an acceleration block and a pair of connecting points (Follower and Base). The details of a body are shown as below:

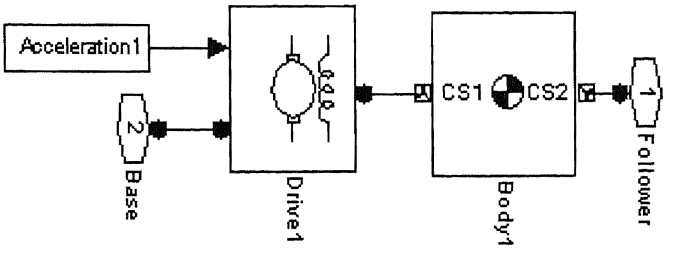

Figure 3.9 Structure of a body

The body block defines the position, weight, and shape information of the corresponding link. The settings of parameters are given in Table 3.4.

Table 3.4 Parameters of the body block

Show Port	Port Side	Name	Origin Position Vector [x y z]	Units	Translated from Origin of	Components in Axes of
☐	Botton	CG	[0 -0.1750+0.25/2 0	m	World	World
☑	Botton	CS1	[0 -0.1750 0]	m	World	World
☑	Top	CS2	[0.075 0 0]	m	World	World
☐	Top	CS3	[-0.075 -0.1750 0.1]	m	CG	CG
☐	Top	CS4	[0.075 -0.1750 0.1]	m	CG	CG
☐	Top	CS5	[0.075 -0.1750 -0.1]	m	CG	CG
☐	Top	CS6	[-0.075 -0.1750 -0.1]	m	CG	CG
☐	Top	CS7	[-0.075 0.075 0.1]	m	CG	CG
☐	Top	CS8	[0.075 0.075 0.1]	m	CG	CG
☐	Top	CS9	[0.075 0.075 -0.1]	m	CG	CG
☐	Top	CS10	[-0.075 0.075 -0.1]	m	CG	CG

Input signals are often connected to the acceleration block. The input signals can be in the form of curve, constant, as well as the signal from controller.

3.2.2 Actuator and friction modelling

Actuator and friction within a driver were also modelled. The actuator contains a rotor, a motor and a gearbox. The settings of parameters are based on Manutec r3 industrial robot arm. The description of Manutec r3 robot arm is given in the previous section of this chapter. The actuator was modelled using an actuator block (known as Motor circuitry bock in SIMULINK) and the friction was modelled using a friction block (known as Coulomb friction block in SIMULINK), as illustrated in Figure 3.10.

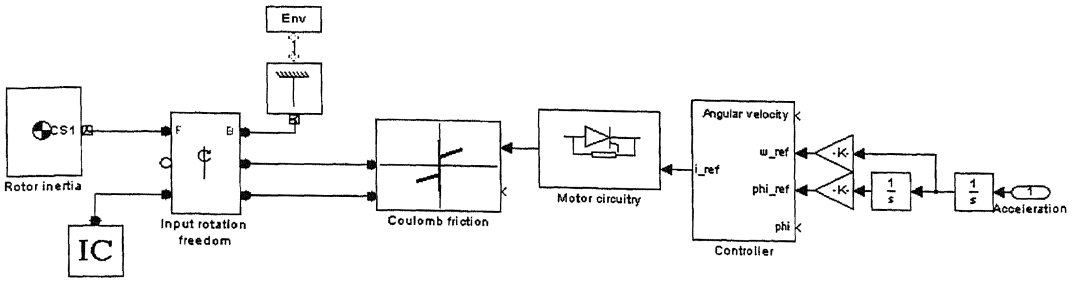

Figure 3.10 Structure of a driver which contains actuator and friction

It can be seen from Figure 3.10 that the actuator model of the robot arm has three main blocks. They are controller block, motor circuitry block, and input rotation freedom block which represent gearbox. Trajectory commands are filtered and converted into control signals (current) by the controller block. The control signals are the inputs to the motor. The gearbox is driven by the motor. The coulomb friction block produces friction of the actuator.

In order to simulate the effect of friction, the coulomb friction block had been added into the actuator module. The joint state changing flow can be illustrated in Figure 3.11.

The variables v and a are the velocity and acceleration along or around a joint primitive axis. These quantities are relative between the two bodies at the joint ends and signed ± to indicate forward or reverse. The joint directionality is set by the base-to-follower sequence of bodies attached to the joint primitive being actuated.

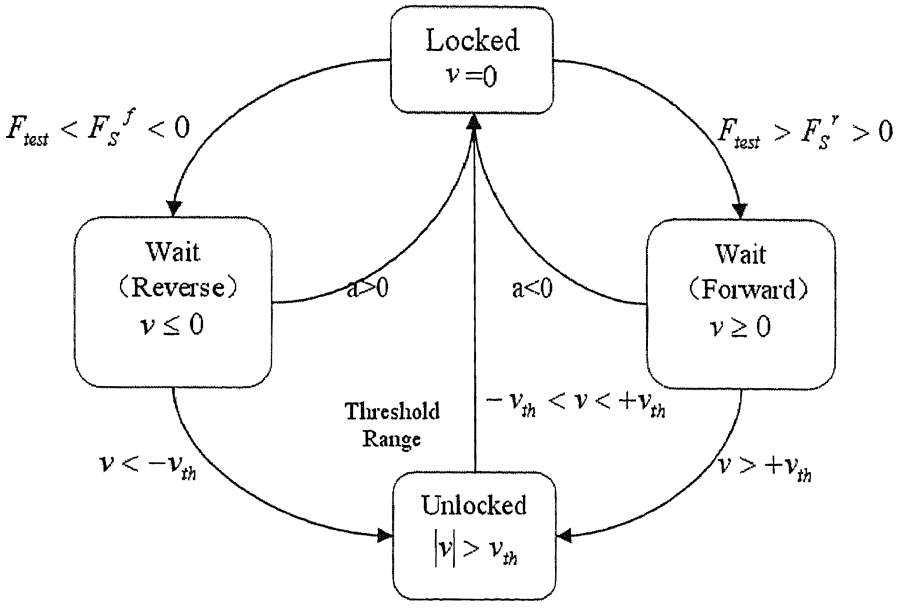

Figure 3.11 State changing flow of a joint

If a joint is moving in continuous motion, during this motion, there are two kinds of torque applied to the joint primitive, they are:

- Kinetic friction torque F_K ($F_K < 0$ retards forward motion, $F_K > 0$ retards reverse motion)
- External, non-frictional torque F_{ext}.

Besides its continuous motion mode, the joint has two other discrete modes, namely, locked and unlocked modes. The coulomb friction block switches a joint primitive between these two modes. In the locked mode, the joint locks rigidly. In the unlocked mode, it moves with the kinetic friction and external non-frictional torques applied. The joint can also be in a wait mode, between the locked and the unlocked modes.

The unlocked mode is specified by a two-condition threshold, they are:

- Joint unlocking threshold velocity $v_{th} > 0$

- Static friction limits $F_S{}^f < 0$ and $F_S{}^r > 0$ for forward and reverse motion.

In the locked mode, v and a of the joint are zero. The static computed torque F_S at the joint is internally computed to maintain the following equitation:

$$F_{ext} + F_S + F_F - F_B = 0 \qquad (3.5)$$

where F_F and F_B are the torques on the base and follower bodies apart from those torques acting at the joint. The joint remains locked as long as $F_S{}^f < F_{test} < F_S{}^r$.

If the static test friction F_{test} leaves the static friction range $[F_S{}^f, F_S{}^r]$, the joint satisfies the first condition for unlocking and enters the wait mode, suspending the mechanical motion. A search begins for a consistent state of the joint in the model. The potential direction of motion after unlocking is determined by all the non-frictional forces on the bodies. During the search, the net torque at the joint primitive is computed by the following equation:

$$F = F_{ext} + F_K \qquad (3.6)$$

where F_K is the kinetic friction.

At this stage a is determined. For potential motion in the forward (reverse) direction, if a<0 (a>0), the search returns to the locked mode. Once a consistent state for the joint is found, mechanical motion restarts. The simulation integrates a to obtain v. When $|v|$ exceeds v_{th}, the joint unlocks.

In the unlocked mode, the joint primitive moves are actuated by the sum of the external, non-frictional torque F_{ext} and the kinetic friction F_K. The wait mode prevents infinite cycling between locked and unlocked modes.

CHAPTER 4. COOPERATION BETWEEN TWO ROBOTS

4.1 Introduction to Fuzzy Logic and Fuzzy Logic Control

The development of Artificial Intelligent enables computers to have certain level of intelligence such as thinking, decision-making, creativity, and the adaptation to complex environments. Fuzzy logic is tightly integrated with the computer science. The fuzzy logic based computer programs which simulate the process of human reasoning, are widely used in various areas, such as automatic tank driving, furnace smelting automatic control, biology, medical diagnosis, economics and social sciences.

Fuzzy logic, also known as fuzzy set theory, differs from traditional set theory. Fuzzy set theory deals with vagueness, a type of uncertainty. For example, between the concepts of "young" and "old", there is no actual criterion from where "young" stops and "old" starts in human reasoning. Zadeh (1965) defined two fuzzy sets for the concepts of "young" and "old", as shown in Figure 4.1. It can be seen that the fuzzy sets allow their elements (age) to partially belong to them and they overlap with one another around the age of 50.

Human being's daily actions can also be vague from human reasoning point of view. For example, when a human picks up a book from a desk, he does not need to calculate a precise force he should apply to the book. The vagueness of the daily actions inspires the development of fuzzy logic control. It has been found that sometimes traditional controllers are difficult to develop in order to control highly

non-linear and highly complex processes. On the other hand, a well trained human operator can finish very complex tasks based on his experience. Engineers can sum up the worker's experience to a group of fuzzy rules and develop a fuzzy logic controller which uses the group of fuzzy rules as a key component. The controller is able to finish complex tasks in the same way that the human operator does.

When a human operator operates a machine, the operator and the machine are connected to form a closed-loop system as illustrated in Figure 4.2. This system can also be called a "human-machine system".

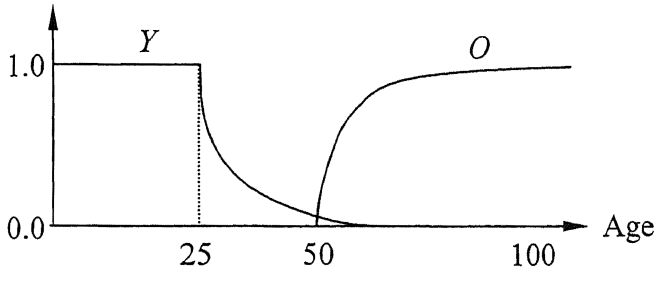

Figure 4.1 Fuzzy sets representing "young" and "old"

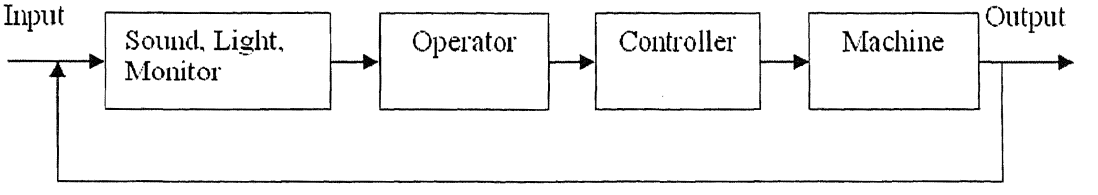

Figure 4.2 A human-machine control system

Firstly, the operator uses eyes and ears to acquire information from the output of the machine in the forms of sound, light and digital/analogous display. The information may include "the pressure is high" or "the changing of temperature is small". He then

converts the information into fuzzy information. Next, the operator uses the fuzzy information and his experience to make control decisions. Apparently, the information in the operator's mind is fuzzy.

The operators experience can be summarised and represented in the form of fuzzy rules. The fuzzy rules can be saved on computers. In addition, human reasoning process can also be simulated using computer programs. Computers, once equipped with the fuzzy rules and the human reasoning programs, can then make fuzzy decisions when provided fuzzy information to achieve a given goal.

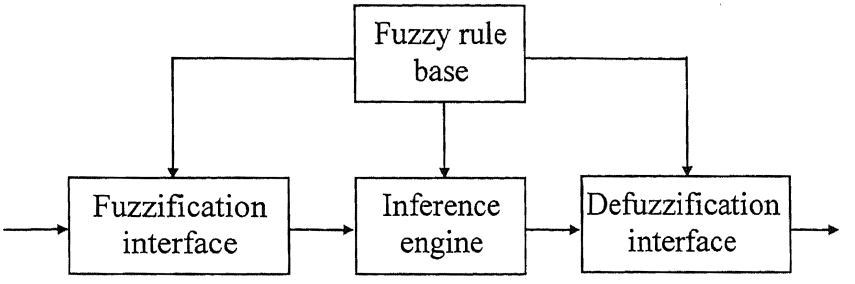

Figure 4.3 Structure of fuzzy logic controller

In Figure 4.3, the fuzzification block represents the process of converting precise input information to fuzzy input information. Whilst, the defuzzification block stands for the process of converting fuzzy decision (output) into precise control signals. Fuzzy rules are stored in the fuzzy rule base. The fuzzy inference engine is a computer program which mimics the process of human reasoning.

Three basic variables are in concern when human operators control a machine. They are errors (e), differences between the desired output and the actual output of the

controlled machine, the changing rate of the differences (c) and the control action (u). The fuzzification block converts the precise values of e and c into fuzzy values. This process is performed based on fuzzy sets which are pre-defined for these two variables. The simplest fuzzification process is called fuzzy singleton, illustrated in Figure 4.4. In the example shown in this diagram, A and B are two fuzzy sets, defined for the variable e. e_p is a precise value of e. Fuzzy singleton converts e_p into $\frac{0.4}{A}+\frac{0.6}{B}$.

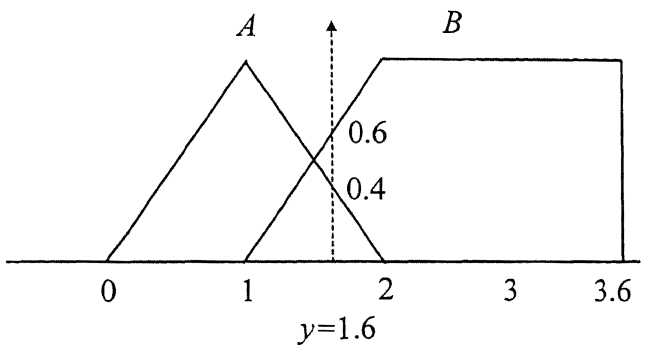

Figure 4.4 Fuzzy singleton

The defuzzification block converts the fuzzy values of u into precise values. The most commonly used defuzzification strategy is Centre of Gravity (CoG). CoG can be formulised below:

$$z_0 = \frac{\sum_{i=1}^{n} \mu(w_i) \cdot w_i}{\sum_{i=1}^{n} \mu(w_i)} \qquad (4.1)$$

where w_j stands for the support value at which the membership function $\mu(w_i)$ reaches the maximum value, and z_0 is the precise value resulted from the defuzzification process.

The fuzzy rule base contains a group of logic rules which are obtained by generalising human operators' experience. Fuzzy rules are normally expressed in the form of "If ...then..."). For example, "If e is A and c is B, then u is C", where A, B and C are three fuzzy sets defined for e, c and u, respectively.

Fuzzy inference is the process of mapping from given inputs to an output based on fuzzy rules. The mapping then provides a basis from which decisions can be made, that is, a control action can be decided. The fuzzy inference engine is an implementation of the fuzzy inference process.

4.2 Fuzzy Logic Controller Design Using MATLAB

4.2.1 Fuzzy logic controller in MATLAB

The Fuzzy Logic Toolbox within MATLAB contains five operations which are needed for implementing a fuzzy logic controller. They are fuzzification, fuzzy composition rule of inference, fuzzy implication, aggregation of the consequents across the rules, and defuzzification.

Fuzzification takes the inputs and determines the membership degrees to which they belong to each of the fuzzy sets defined for the inputs via membership functions. It gives out a set of degrees of membership in [0, 1] of the corresponding fuzzy sets. The fuzzification process is illustrated in Figure 4.5.

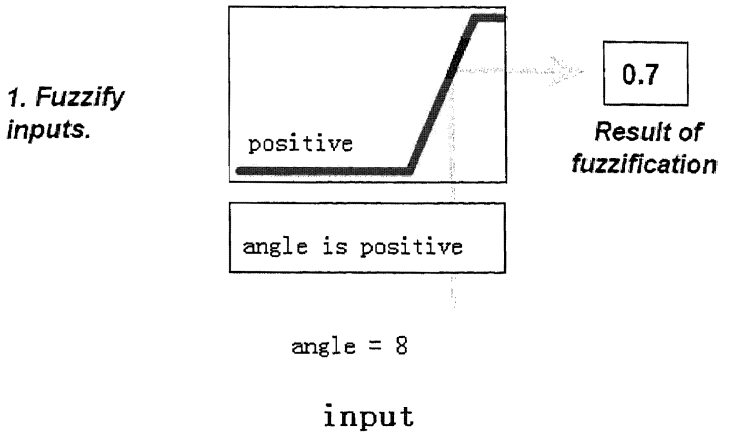

Figure 4.5 Fuzzification process in MATLAB

Mamdani-type fuzzy inference is the most commonly used fuzzy inference method in automatic control. It was developed when Mamdani (1974) designed the first fuzzy logic control system to control a steam engine and boiler combination. In Mamdani-type fuzzy inference, the composition rule of inference is implemented using "super-min" to activate fuzzy rules, fuzzy AND operator is then applied to implement fuzzy implication, also known as fuzzy mapping, and finally fuzzy OR operator is applied to the consequence of the rules for aggregation. In the Fuzzy Logic Toolbox, because of the use of fuzzy singleton, "super-min" becomes simple and straightforward. Fuzzy AND is implemented by point-wise minimum and fuzzy OR by point-wise maximum. Mamdani-type fuzzy inference can be illustrated in Figures 4.6 and 4.7.

Defuzzification strategy used in the Fuzzy Logic Toolbox is the Centre of Gravity, which returns the centre of area covered by a fuzzy set. This process is illustrated in Figure 4.8.

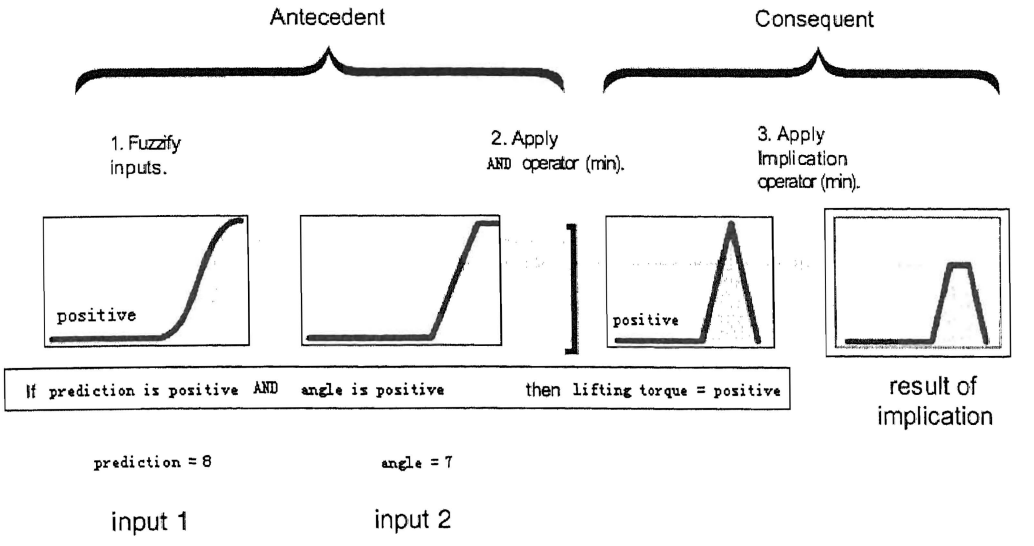

Figure 4.6 Mamdani-type fuzzy inference

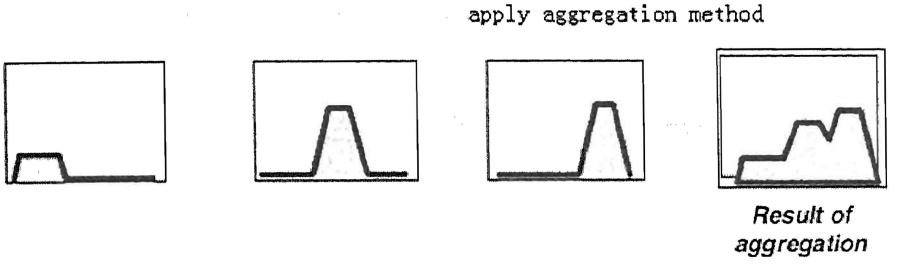

Figure 4.7 Aggregation process

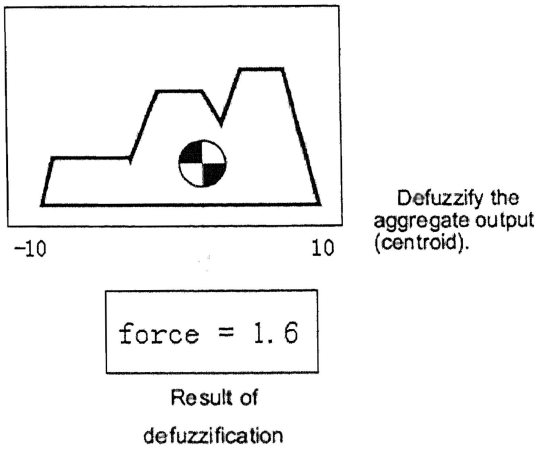

Figure 4.8 Defuzzification process

4.2.2 Fuzzy logic toolbox

The Fuzzy Logic Toolbox provides graphical user interface (GUI) tools for the design of fuzzy logic controller. The followings are five primary GUI tools for building, editing, and viewing a fuzzy logic controller:

- Fuzzy Inference System (FIS) Editor
- Membership Function Editor
- Rule Editor
- Rule Viewer
- Surface Viewer

These GUI tools are interconnected. If any change is made to a fuzzy logic controller through one of them, all other GUI tools will make the corresponding changes. The connections among the GUI tools are illustrated in Figure 4.9.

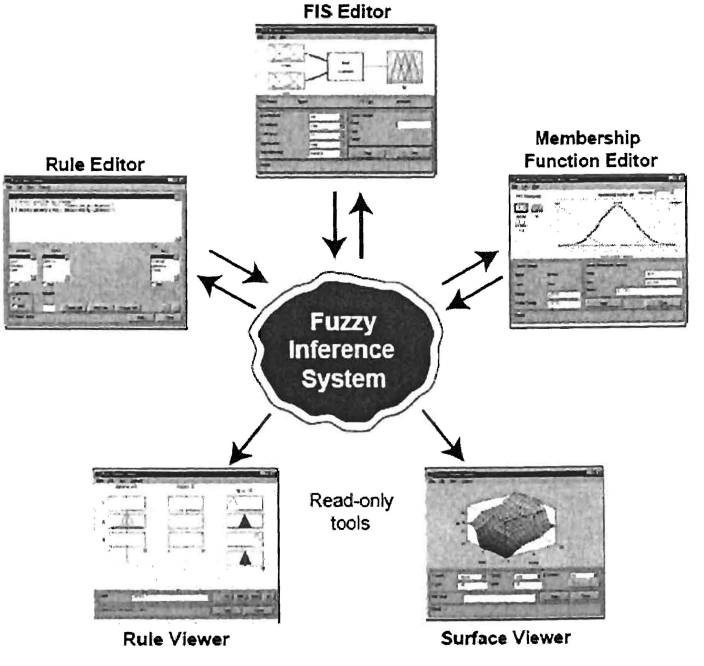

Figure 4.9 Interconnected primary GUI tools in Fuzzy Logic Toolbox

The names and the number of input/output variables of a fuzzy logic controller are defined in the FIS Editor. The Membership Function Editor is used to define membership functions of all fuzzy sets associated with each variable. The Rule Editor is for editing fuzzy rules. The Rule Viewer and the Surface Viewer are used to display all fuzzy rules in either "If...Then..." format or graphically. They are strictly read-only tools.

The five primary GUI tools can all interact and exchange information. Any one of them can read from and write to the workspace and to a file (the read-only viewers can still exchange plots with the workspace and save them to a file). For any fuzzy inference system, any or all of these five GUI tools may be open. If more than one of these editors is open for a single system, the various GUI windows are aware of the existence of the others, and, if necessary, update the related windows. Thus, if the names of the membership functions are changed using the Membership Function Editor, those changes will be reflected in the rules shown in the Rule Editor. The editors for any number of different FIS systems may be open simultaneously. The FIS Editor, the Membership Function Editor, and the Rule Editor can all read and modify the FIS data, but the Rule Viewer and the Surface Viewer do not modify the FIS data in any way.

4.3 Development of Fuzzy Logic Controllers for Robot Arm Cooperation

The aim of developing fuzzy logic controllers for a robot arm is to enable it to cooperate with a human in lifting a light-weight stick and keeping the stick level

during the course of lifting. The stick is light weight and flexible so force control is no considered to the end-effectors. The two robot arms each of which holds one end of a stick are illustrated in Figure 4.10.

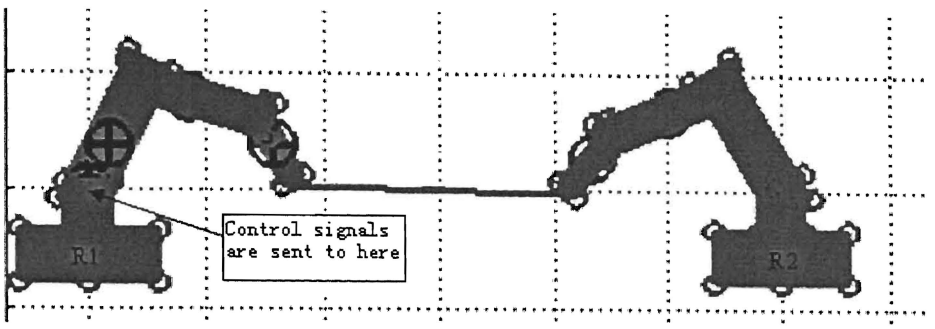

Figure 4.10 Two robot arms holding a stick

Fuzzy logic controllers were designed for robot arm on the left, R_1. Robot arm, R_2, is a normal industrial robot arm the motion of which follows a pre-defined trajectory. Two fuzzy logic controllers were developed for R_1. One is for cooperating with R_2 in lifting the stick, and the other is for keeping the stick level when R_2 stops moving.

4.3.1 Designing fuzzy logic controller to following the motion of R_2

The fuzzy logic controller for R_1 to follow R_2's motion has two inputs. One is the angle between the stick and the ground, as this angle tells whether R_2's motion is followed. The other is the change rate of R_2's lifting speed. This signal helps R_1 to predict R_2's next movement. A positive value of this signal indicates that R_2 will continually lift the stick, whilst a negative value means R_2 is going to stop.

Assigning the robot arms a reference system, as displayed in Figure 4.11, the height of the end-effector of R_1, y_1, and that of R_2, y_2, can be measured. The position of the end-effector of R_1 along X-axis, x_1, as well as that of R_2, x_2, can also be measured. The angle θ, can then be calculate as the following:

$$\theta = \arctan\frac{y_1 - y_2}{x_1 - x_2} \tag{4.2}$$

This equation can also be implemented in SIMULINK in the way shown in Figure 4.12.

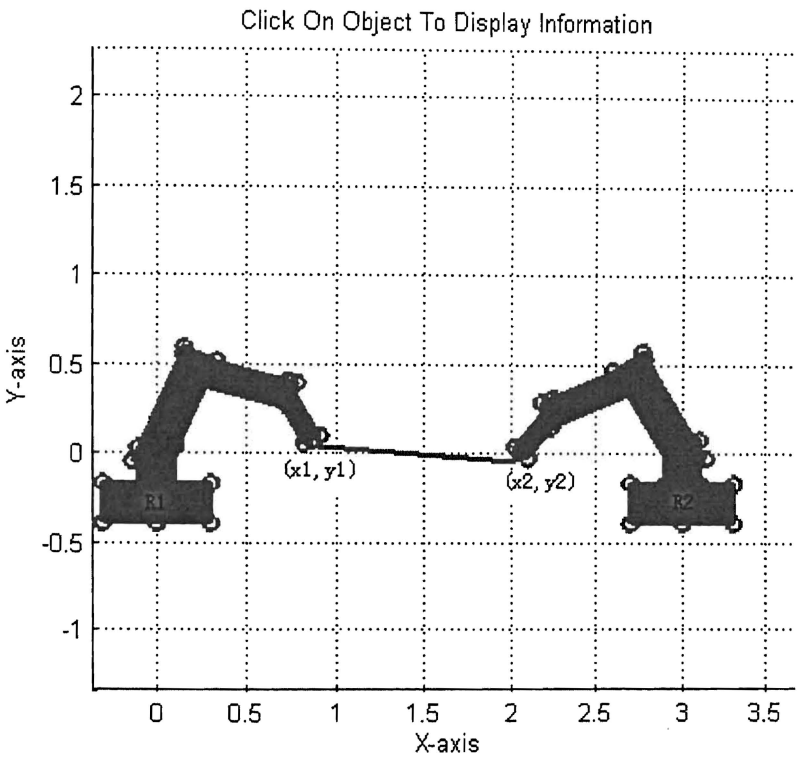

Figure 4.11 Reference system assigned to the robot arms

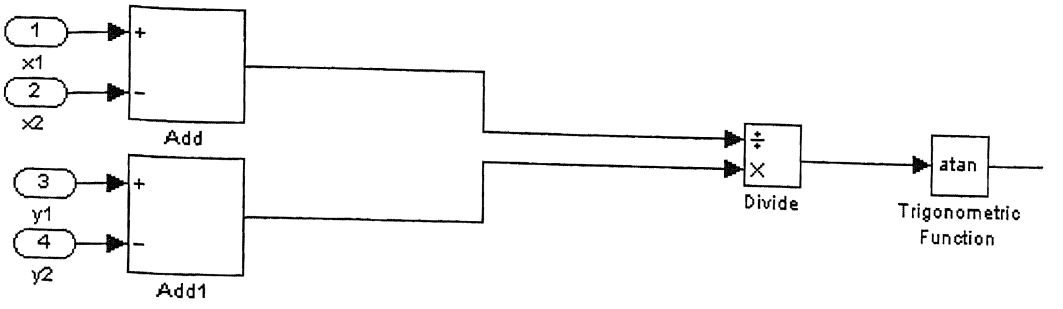

Figure 4.12 Calculation of the angle θ in SIMULINK

The change rate of R_2's lifting speed, Δv, can be calculated using the following equation:

$$\Delta v = \frac{dY_2(t)}{dt} - \frac{dY_2(t-\Delta t)}{dt} \quad (\Delta t = 0.1s) \tag{4.3}$$

This equation can also be implemented using SIMULINK, as illustrated in Figure 4.13.

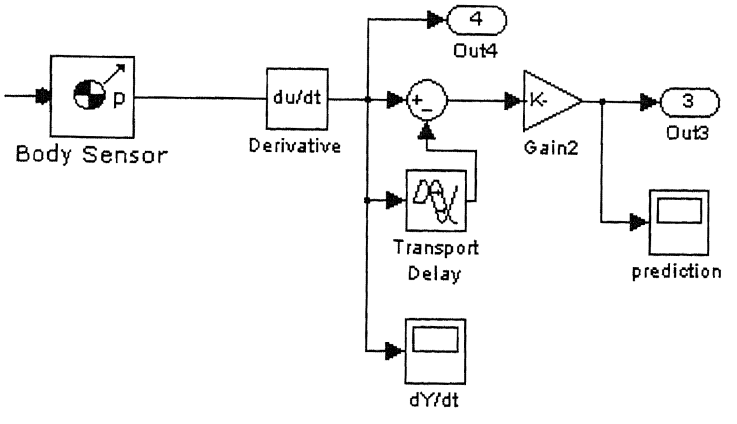

Figure 4.13 Calculation of the change rate of R_2's lifting speed in SIMULINK

The membership functions of these inputs are defined based on experience and simulation. The membership function of the first input "angle" is given in Figure 4.14.

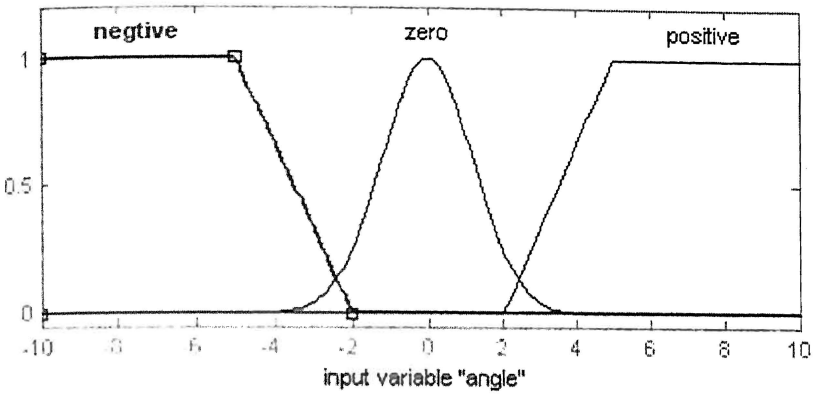

Figure 4.14 Membership function of "angle"

In order to let the R_1 respond quickly, the membership function for "negative" and "positive" fuzzy sets were defined having the shape of "trapezoid" and that for the 'zero' fuzzy set was in Gaussian shape.

The membership function of the input "the change rate of lifting speed" is shown in Figure 4.15. The membership functions for all fuzzy sets are in Gaussian shape.

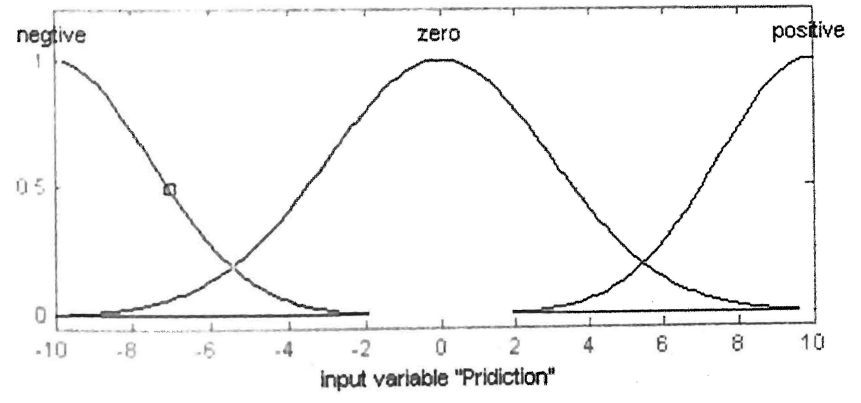

Figure 4.15 Membership function of "change rate of lifting speed"

The design of fuzzy rules is based on experience. For example, if the end-effector of R_2 is higher than that of R_1 and the change rate of R_2's lifting speed decreases, then the controller should keep the current lifting torque and wait. The fuzzy rule base of the fuzzy logic controller is given in Figure 4.16. It can also be viewed in the input-output space, as shown in Figure 4.17.

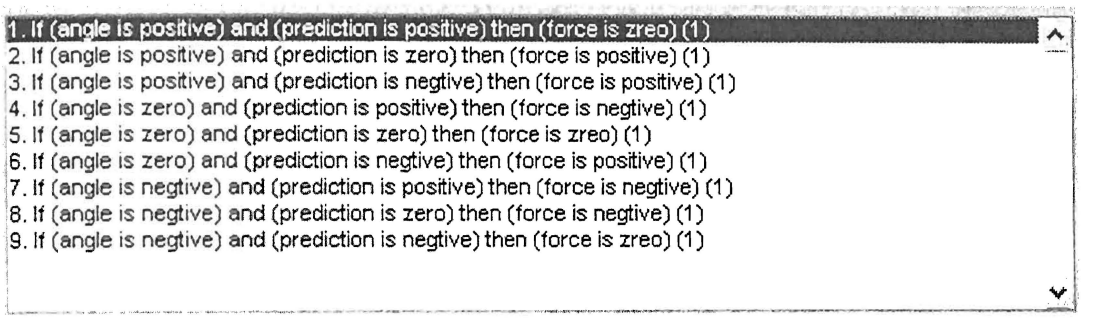

Figure 4.16 Fuzzy rule base

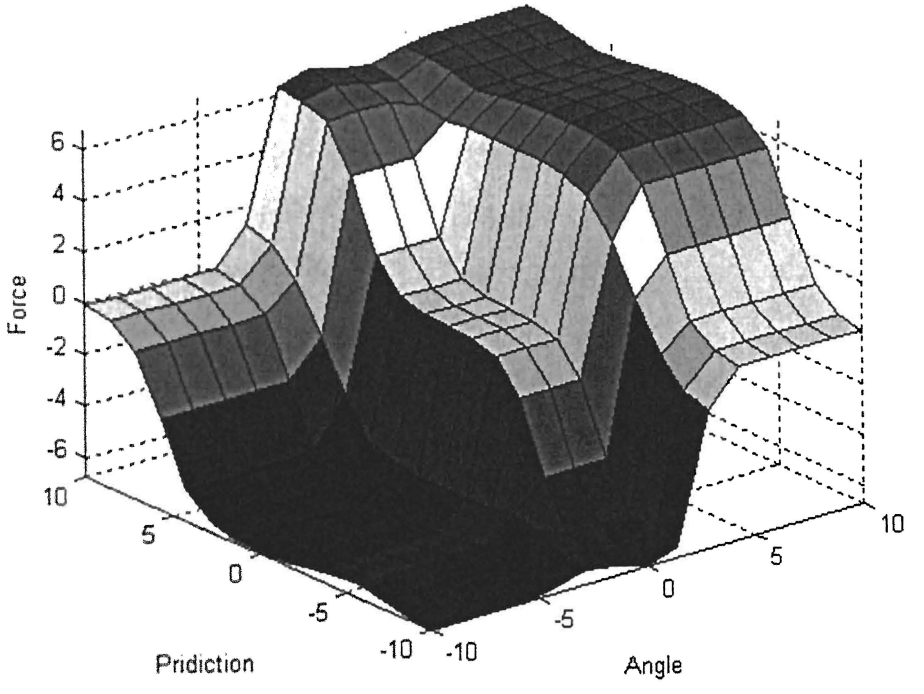

Figure 4.17 Mapping from inputs to output

These two inputs are fed into a fuzzy inference engine which is provided by the Fuzzy Logic Toolbox. Based on the fuzzy rules, the inference engine can produce a lifting torque which is the output of the controller. The fuzzy sets of this output are defined as shown in Figure 4.18.

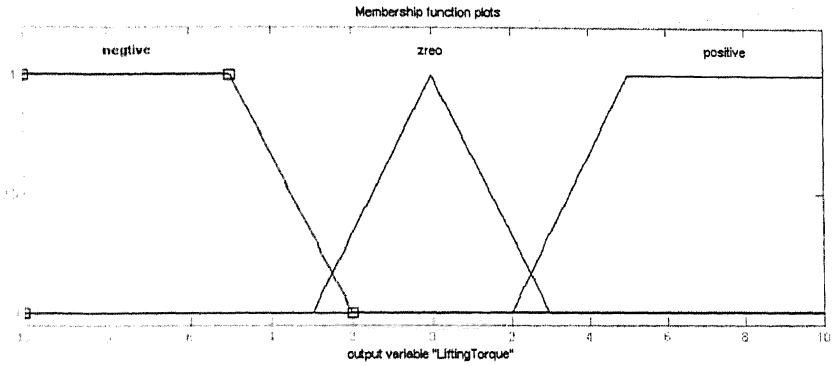

Figure 4.18 Membership function of "lifting torque"

The controller can now be shown in Figure 4.19.

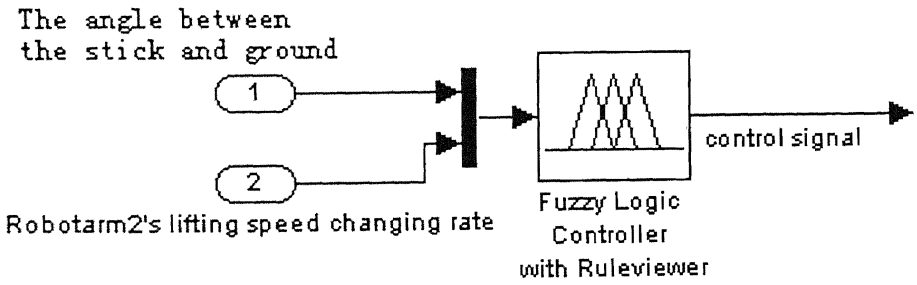

Figure 4.19 Fuzzy logic controller for R_1 to follow the motion of R_2

As described in the last chapter, the robot arm has four links. In order to get best control results, all the control signals are sent to the actuator between link1 and link2

(as can be seen from Figure 4.10). The output of the controllers will be transformed to current signals and applied directly to the joint between the two links.

4.3.2 Designing fuzzy logic controller for keeping the stick level

In order to keep the stick level after R_2 stops moving, an additional fuzzy logic controller was added for R_1. This controller also has two inputs and a single output. The two inputs of this controller are the angle θ and the R_1's lifting speed. The output is the lifting torque.

The fuzzy rules of this additional controller are given in Figure 4.20. The fuzzy rules were also obtained based on experience. For example, when the angle is positive and the lifting speed of R_1 is also positive, that means the stick will tend to level, then R_1 should keep the current torque and wait.

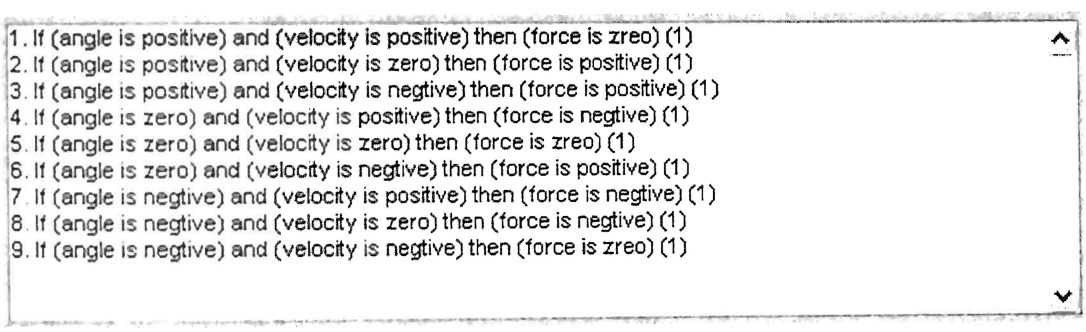

Figure 4.20 Fuzzy rules developed for the additional controller

The mapping of the controller's inputs and output is illustrated in Figure 4.21.

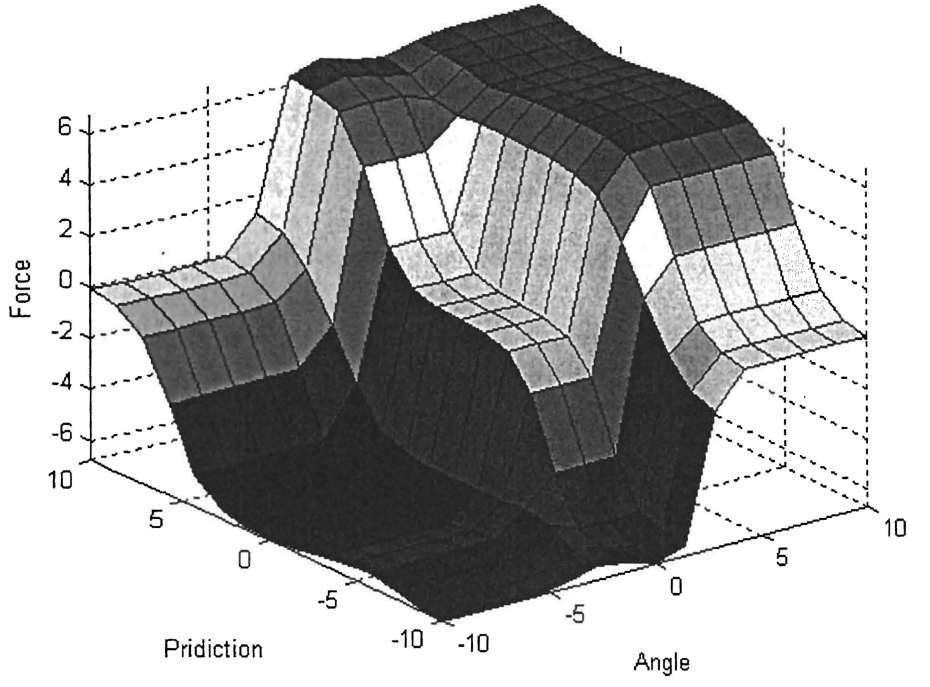

Figure 4.21 Mapping of the inputs to the output of the additional controller

A switcher is designed to switch between the two fuzzy logic controllers. The switch normally switches on the first controller. When the switcher detected no further movement of R_2, which is done by measuring the height of the end-effector of R_2, then the switch switches the additional controller. The switcher is illustrated in Figure 4.22.

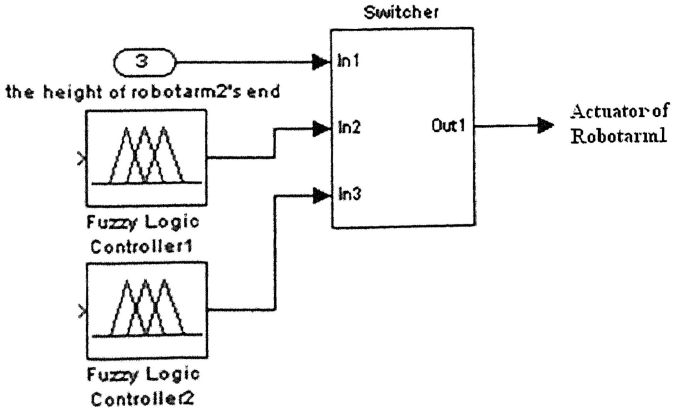

Figure 4.22 The switcher to switch on controller2

4.4 Simulation Results

R_2's motion trajectory is shown in Figure 4.23. X axis represents time and Y axis represents the height of the end-effector of R_2. The robot arm lifted the stick to the height of $0.5m$ in $2.5s$. R_1 was controlled by the fuzzy logic controllers. R_1's movement is illustrated in Figure 4.24. In this diagram, X axis represents time and Y axis represents the height of the end-effector of the robot. An overshot can be observed from Figure 4.24.

The changes in the angle between the stick and the ground are given in Figure 4.25. Y axis represents the angle. It can be seen from Figure 4.25, the maximum of the angle is $0.0741rad$ (approximately 4.25°). Because of the effect of the additional controller, the angle was reduced to $0.0235rad$ (approximately 1.35°) after R_2 stopped moving. The stick was almost kept level during the whole process of lifting. The time between the R_2 stopped moving and the stick was kept level is $1.5s$.

The result also shows that when R_2 lifted the stick with a speed of $0.3m/s$, R_1 controlled by the first fuzzy logic controller can response quickly.

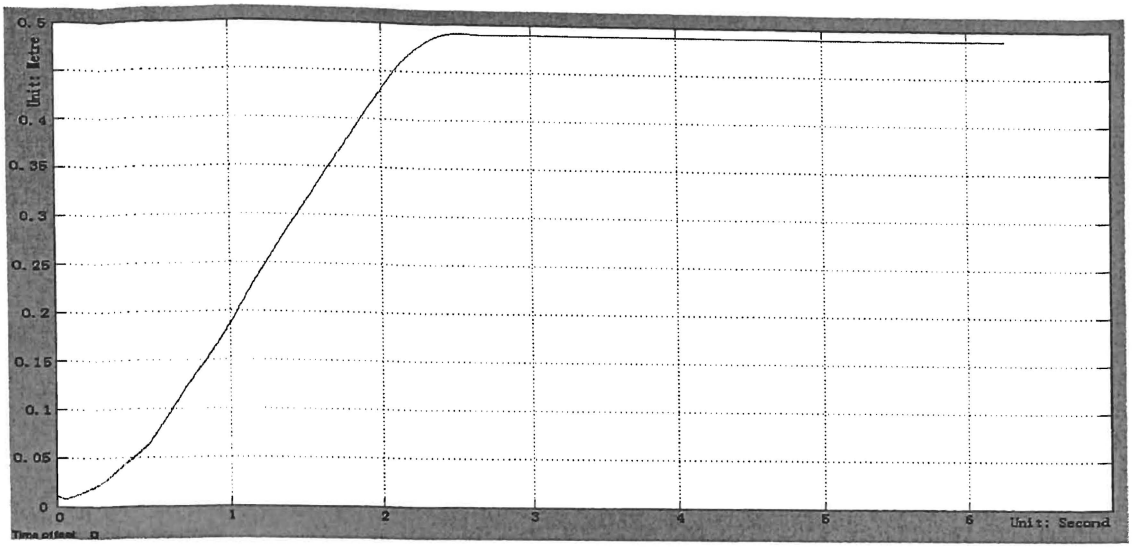

Figure 4.23 The height of R_2's end-effector

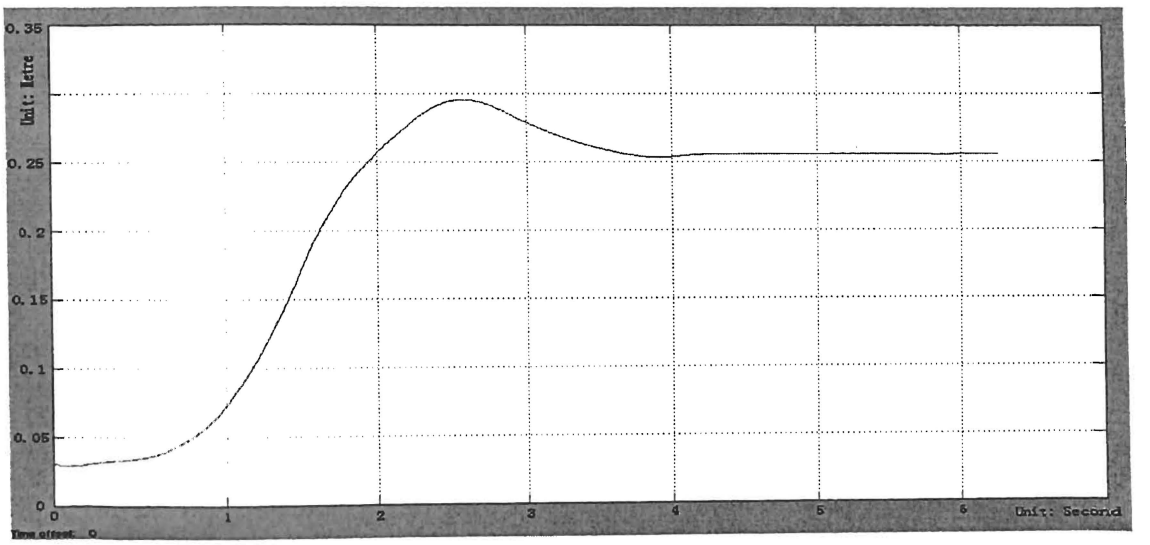

Figure 4.24 The height of R_1's end-effector

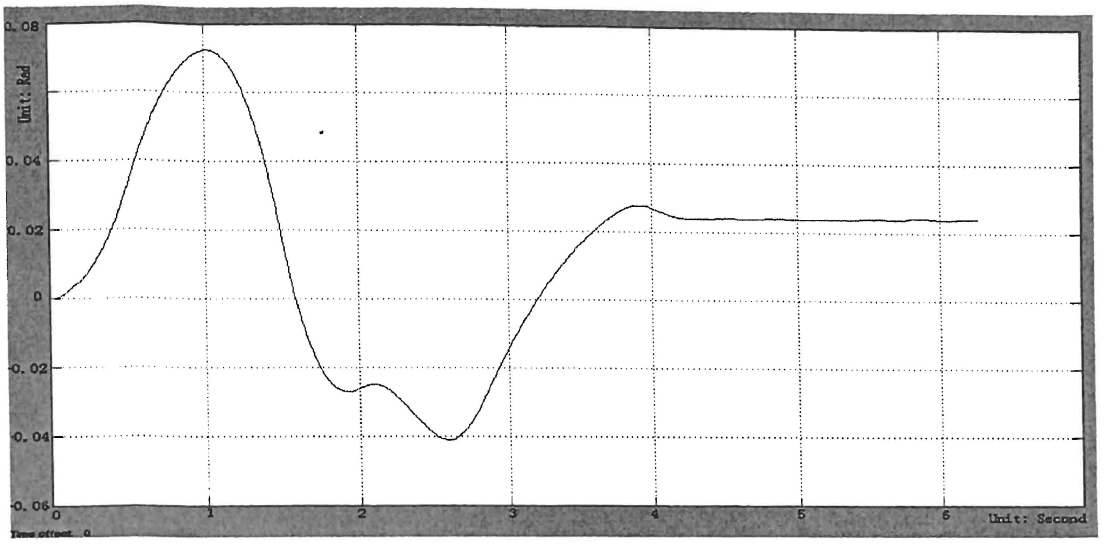

Figure 4.25 Changes in the angle between the stick and the ground

CHAPTER 5. FRAMEWORK OF ACTIVE ROBOT LEARNING AND COOPERATION WITH ACTIVE COGNITIVE ROBOT

In order to enable a robot to cooperate better with its user, the robot needs to be able to predict what the user will do next. The prediction can be made based on the users' intention and/or preference, that is, the high-order beliefs of the robot. The existing approaches to the development of such beliefs through machine learning rely on particular social cues or specifically defined award functions, as mentioned in Chapter 2. Their applications can be limited.

This chapter presents an active robot learning (ARL) approach for a service robot to develop such beliefs. Inspired by discovery learning theory which encourage learners to acquire information by performing their own experiments, this approach allows a robot to perform guided tests on its users and to build up the high-order beliefs according to the users' responses. This approach emphasises the active acquisition of intention and preference by robots but not the passive learning. The robots do not require recognition of a particular gesture or determination of a specific function.

5.1 Robot Active Learning

Discovery learning takes place in problem solving situations where the learner draws on his own experience and prior knowledge. It has then been developed into a method

of instruction through which learners interact with their environment by exploring and manipulating objects, wrestling with questions and controversies, or performing experiments.

Discovery learning can be simply described as "learn by doing". Despite of concerns to its effectiveness, this learning method has been employed in supervised machine learning , known as active machine learning (AML), as a resolution to the problem of lacking expensive labelled training examples. Supervised learning requires a sufficient number of labelled training examples to be presented to the learner. An error signal between the examples and the learning outcomes from the learner can then be obtained and used to drive the learning process, for example, through adjusting the learner's parameters to minimise the error. The labelled data, however, are sometimes expensive to obtain. AML allows the learner to explore all available examples and to add scores to them. Those with higher scores will be passed to human experts to add labels (Kim, et al., 2006). AML has also been used in robot control to model the inverse dynamics of a robot arm with high model uncertainty (Robbel and Vijayakunar, 2007).

Uncertainty also exists in the development of beliefs for service robots on their users' intentions and preferences. Belief is the psychological state in which an individual (including cognitive robots) holds a proposition to be true. In computer science, the decision on whether the proposition is true (uncertainty) can be made by looking at the evidence of other related propositions (Dempster 1968, Shafer 1976). The collection of relevant evidence is, therefore, an important step in the process of building up beliefs. In situations where service robots co-work with their users, to

build up their high-order beliefs on the users' intention and preference, the robots also need to collect evidence which may not be seen at the first glance.

The property of "learning by doing" of discovery learning makes it suitable for the robots to develop the high-order beliefs. With discovery learning capability, the robots will be able to perform experiments when they are not sure what their human counterparts intend and prefer to do. By doing this, the robots can discover evidence which is required but not seen at the first glance to build up the high-order beliefs. For example, in the situation where a robot helps its user to lift an object from ground and the robot realizes that the user stopped at certain height, the robot will need to find out whether the user decides to lift the object only to that height or he prefers to put the object down to the ground because he changes his mind. The robot can test the user by slightly putting down the object and see how the user response. If the robot perceives the same action from the user, it can then regard the response as the evidence of changing mind. If the robot perceives no action from the user, it can view this response as evidence of the preference of keeping the object to that height.

This approach to the use of discovery learning in the development of service robots' beliefs can be called active robot learning (ARL). ARL differs from AML because ARL requires a robot to carry out experiments to generate data (evidence), whilst AML only searches for and evaluates available data.

5.2 Framework of ARL System

The overall structure of an ARL system is shown in Figure 5.1. The system consists of an action bank which stores actions that can be taken to test its users, an inference engine which reasons about what actions to be taken for a specific purpose, a moment determination mechanism to decide the moment of test, an intention identification mechanism to interpret responses of the users and to identify intention and preference, and an intention model which represents intentions.

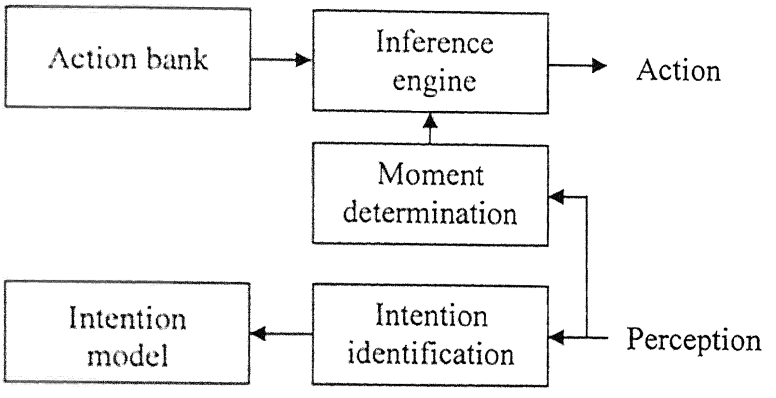

Figure 5.1 Structure of ARL system

5.2.1 Action and action selection

Test actions are those that can be taken to test the users. They are associated with conditions and stored in the action bank. Each test action stored in the action bank has a name and content which is the kinematics of the robot. The conditions express reasons for performing the actions and are represented as propositions. For example, if a robot hands over a glass of water to its user, it would need to check whether the user intends and is ready to take over the glass. The test action for testing the user in

this case is to slightly loosen the glass and the condition associated is to confirm the user's intention of taking over the glass. The actions and the associated conditions can be designed by robot designers before the robots are deployed.

The inference engine selects a test action from the action bank to conduct a specific test. As the actions are associated with conditions in the action bank and the associations actually represent causal relations (implications) from the conditions to the actions, the selection of an action can be carried out with the standard forward reasoning.

The moment determination mechanism decides the starting time for testing the user and triggers the action bank to send out a test action. There are, in general, two moments where a robot needs to test the user for intentions. The first is the moment before the last action in the course of the completion of a task. Taking the example of getting a drink for the user, before a robot finally takes the action of releasing a glass, it needs to find out whether the user intends and is ready to take over the glass. In the example of assisting a human standing out from a chair, the robot has to make sure that the user intends and is ready to stand alone before releasing his arm/hand. The second is the moment when a robot feels its user stops doing what he originally intended to do. The robot will need to find out whether the user changed his mind or not for further cooperating with him. In this situation, the robot will rely on its perception to detect this stop.

5.2.2 Fuzzy intention model and intention identification

Human intention and preference can be vague. Human beings can desperately want to do or not to do something and can also more or less want to do it. They can like something very much, more or less like, less like, or dislike it. To reflect this natural vagueness, intentions and preferences are represented using fuzzy sets in the intention model. The fuzzy sets are defined in the domain of intention and preference. Representing intention as a variable, linguistic terms used by human beings to describe their intentions, for example, can be discrete values of this variable and arranged along the axis of intention from "determinatively to do", "may want to do", ... to "determinatively no to do". Fuzzy sets on this axis are defined by membership functions which take values in [0, 1]. For example, "determinatively to do" and "may want to do" can be defined as shown in Figures 5.2(a) and (b), respectively.

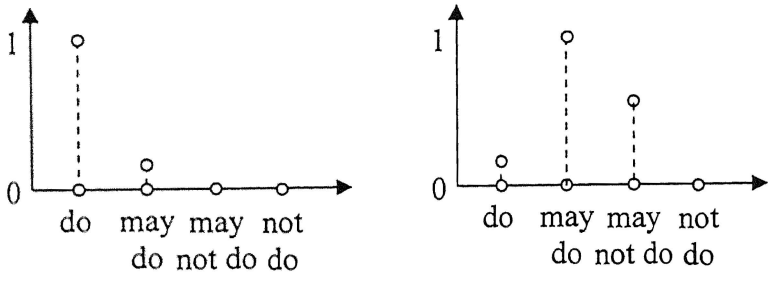

Figure 5.2 Fuzzy sets defined for representing intentions

The vagueness is represented by the membership functions. First, the membership functions can take values in [0, 1]. These values are the subjective expressions about uncertainty property of intention and preference. Second, these functions are allowed to have any shape to represent slight different subjective feelings. For example, the

fuzzy set shown in Figure 5.2(b) represents the hesitation between "do" and "not do" but incline more to "do" side. If the shape changes in the way of increasing the value of "may not do" to 1, the fuzzy set will still express hesitation but much less inclination.

Intention and preference are identified from the responses of the user to the test actions a robot preformed. In this sense, the identification process maps the perception of the robot to the intention and preference of its user, and, therefore, relies on a set of rules which represent the relationship between the perception and intention and preference. It is more natural to develop a causal relation from what the user intends or prefers to do to how the user responds to test actions, that is, from intention and preference to perception, than the way around. This is because that the responses are naturally determined by the intentions and preferences. Each of such rules is also associated with a truth degree indicating to what extend the designers of the rule trusts the rule.

To identify intentions and preferences based on this type of causal relation will need fuzzy backward reasoning (Anould and Tano, 1995; Pham and Li, 2001). Fuzzy backward reasoning performs based on a fuzzy relational equation, the general form of which is represented in 5.1.

$$X \circ R = Y \qquad (5.1)$$

where X is a fuzzy decision variable, Y is s fuzzy dependent variable, R stands for a causal relation from the universe of discourse where X is defined to the universe of discourse where Y is defined, and $\circ$ represents the composition rule of inference.

Because R represents a causal mapping from X to Y, given Y_d, a desired value of Y, fuzzy backward reasoning can deduce X which is sufficient for Y_d. In the case of intention and preference identification, a fuzzy relational equation can be developed to represent the causal relation from intention and/or preference to actions. This is because the intention leads to actions when the user is conscious. When the response of the user to a test action is known, fuzzy backward reasoning is able to deduce the intention or preference that could determine the response by solving 5.1. The user's responses are perceived by a robot in terms of perception.

5.3 Simulations and Discussions

Simulations were designed and carried out to evaluate the ARL based approach to building up high-order beliefs for a robot arm. The simulation environment is depicted in Figure 5.3. Robot arm R_1 is a cognitive robot with the ARL framework installed. Robot arm R_2 models the user. Figure 5.4 shows how the ARL system is integrated with the robot control system described in Chapter 4.

In the simulations, the user and the robot cooperate in lifting a stick in the way of keeping it level. The user then stopped. When the robot detected the stop, it reacted in order to keep the stick level and applied test actions to the user to check his intention in order to further cooperate with him. The user reacted to the test actions taken by the robot. The robot then started to identify the user's intention according to his responses.

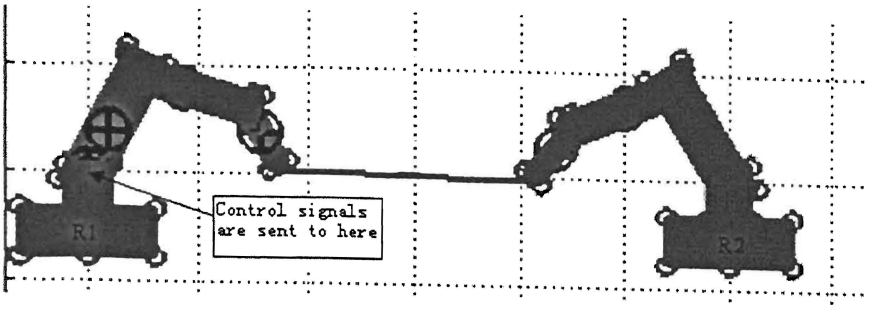

Figure 5.3 Simulation environment

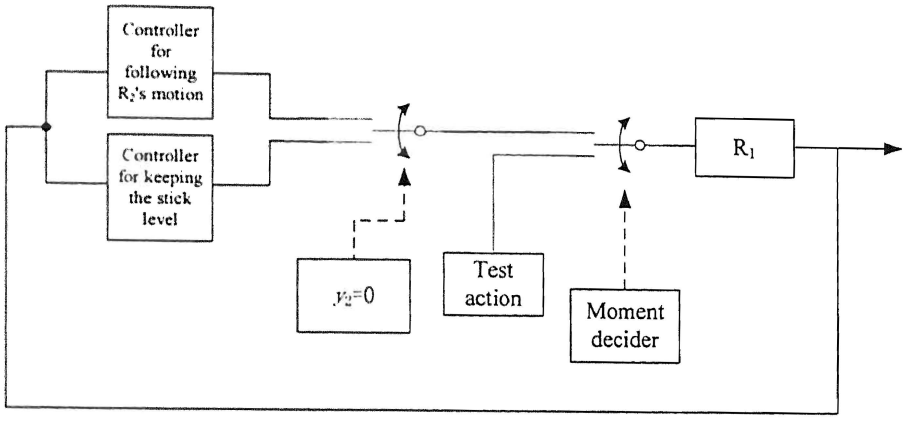

Figure 5.4 Integration of ARL system to robot control system

The user may stop for three different reasons: having a rest (because the stick is heavy), or leaving the stick at the height (because the stick reached the desired height), or putting the stick down (because he changes his mind). Three linguistic terms were used to describe these three reasons, namely, "having a break", "leaving to the desired height", and "putting down". These terms were represented using *a*, *b* and *c*, respectively.

A variable *x* was defined in the domain of intention and *a*, *b*, and *c* are the three values of *x*. Intentions of "to have a rest and then continue", "to keep at the current

height", and "to put down" were defined, in the form of fuzzy sets A, B, and C, respectively, in the domain of x. Human beings can be indeterministic and even hesitate by their nature. That is, vagueness can exist in their intentions. The best way to represent this natural vagueness in human beings' mind is to use fuzzy sets. The fuzzy sets A, B and C are shown in Figures 5.5(a), (b) and (c).

Test actions and associated conditions are:

- If to confirm "continue to lift", Then Lift_slightly
- If to confirm "the desired height reached", Then Stay_still
- If to confirm "put down, Then Put_down_slightly.

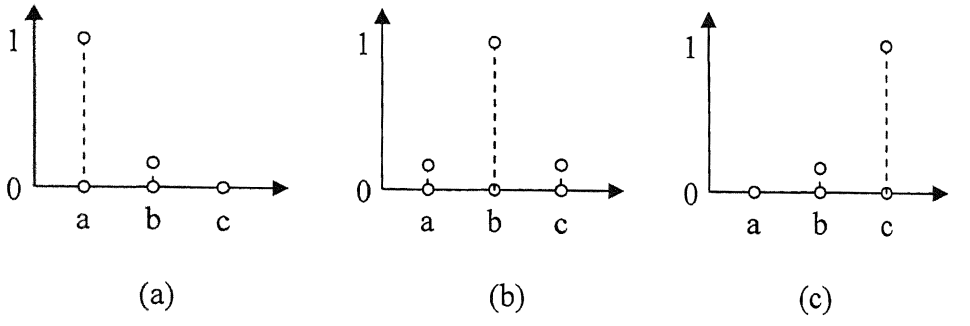

Figure 5.5 Intentions defined as fuzzy sets

A tilt sensor was attached to the stick and connected to the robot for it to check whether the stick is level. The sensor gives a 0 (Z) when the stick is at level, that is, the two sides of the stick held by the user and the robot, respectively, are at the same height. It gives a positive number (P) when the side held by the user is higher than the other side. Otherwise, it gives a negative number (N). The sensor readings in the course of a test provide information about the user's reactions. For example, that the sensor reading is still 0 after the robot took Lift action means the user was also lifting with the robot.

A fuzzy variable X was defined for the user's intention, a "crisp" variable u defined for test actions taken by the robot, and Δy for the differential of the sensor reading. Rules were designed for the robot to reason about the user's intention according to the actions it takes and perception it receives.

In the case where the user intends to have a rest and then continue, if the robot takes action of Lift_slightly, then the user may keep in lifting the stick, though reluctant. This can be seen from Figure 5.5 (a) where both a and b are non-zero. If the robot takes the action of Put_down_slightly, the user will not follow the robot putting down the stick. If the robot takes the action of Stay_still, the user will be happy to remain unchanged. Therefore, the corresponding rules defined are:

- IF X is A AND u is Lift_slightly, THEN Δy is Z (truth degree = 1.0)
- IF X is A AND u is Stay_still, THEN Δy is Z (truth degree = 0.2)
- IF X is A AND u is Put_down_slightly, THEN Δy is P (truth degree = 1.0).

In the case that the user intends to keep the stick at the current height, if the robot takes action of Lift_slightly, then the user will remain his current position unchanged. If the robot takes the action of Stay_still, the user will be happy to remain unchanged. If the robot takes the action of Put_down_slightly, the user will not follow the robot putting down the stick. Therefore, the corresponding rules are:

- IF X is B AND u is Lift_slightly, THEN Δy is N (truth degree = 0.4)
- IF X is B AND u is Stay_still, THEN Δy is Z (truth degree = 1.0)
- IF X is B AND u is Put_down_slightly, THEN Δy is P (truth degree = 0.5).

In the case that the user intends to put the stick down, if the robot takes action of Lift_slightly, then the user will put down the stick. If the robot takes the action of Stay_still, the user will also put down the stick. If the robot takes the action of Put_down_slightly, the user will be happy to cooperate with the robot. Therefore, the corresponding rules are:

- IF X is C AND u is Lift_slightly, THEN Δy is N (truth degree = 0.6)
- IF X is C AND u is Stay_still, THEN Δy is N (truth degree = 1.0)
- IF X is C AND u is Put_down_slightly, THEN Δy is Z (truth degree = 1.0).

The above rules were represented using three fuzzy relational matrices, namely, R^{Lift}, R^{Stay_still} and R^{Put_down}, which correspond to the conditions of taking actions of Lift_slightly, Stay_still and Put_down_slightly, respectively. The fuzzy relational matrices are:

$$R^{Lift} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0.4 \\ 0 & 0 & 0.6 \end{pmatrix} \tag{5.2}$$

$$R^{Stay\_still} = \begin{pmatrix} 0 & 0.2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

$$R^{Put\_down} = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tag{5.4}$$

The variables X and ΔY (ΔY is the set of differentials of sensor readings the elements of which are the truth values of P, Z, and N) can be connected with the fuzzy relational matrices to form three fuzzy relational equations such as:

$$X \circ R = \Delta Y \tag{5.5}$$

where R stands for one of the three matrices given in 5.2, 5.3 and 5.4.

In the first simulation, a 3.5 seconds halt was introduced into the trajectory designed for R_2 to simulate the case where the user takes a rest in the middle of lifting. When R_1 captured the halt through the tilt sensor (Δy remained at 0 in this case), the ARL system applied the test action of Lift_slightly to the stick to test R_2. Because R_2 intended to have a rest and then continue, it responded to R_1 by restarting to lift the stick. When received this response, R_1 started fuzzy backward reasoning to identify R_2's intention, which is "to have a rest and then continue" in this case.

The pre-defined motion trajectory for R_2 is shown in Figure 5.6. X axis represents time and Y axis represents the height of the end-effector. From 0.0s to1.5s, R_2 lifted the stick from the ground up to 0.23 metres. From 1.5s to 5.0s, R_2 stopped to have a rest. From 5.2s to 6.5s, R_2 continued to lift the stick from the height of 0.23m to 0.39m. After 6.5s, R_2 stopped.

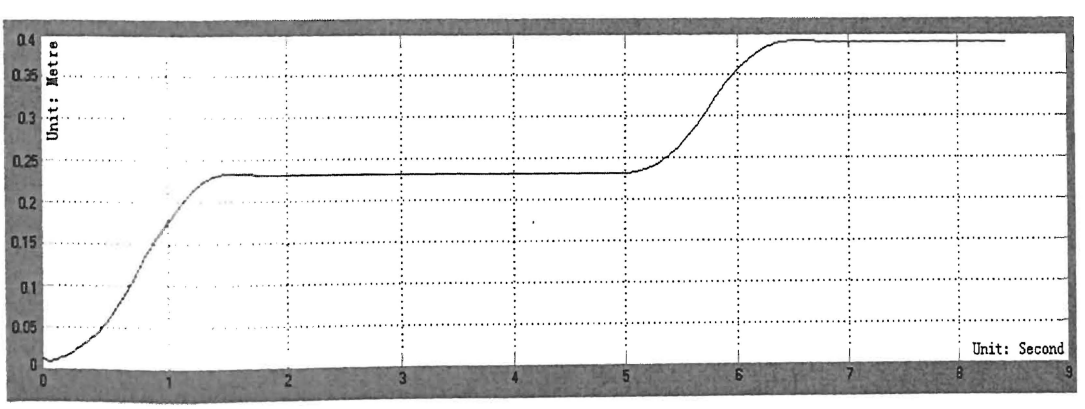

Figure 5.6 Trajectory of R_2's end-effector

R_1's motion trajectory is shown in Figure 5.7. X axis represents time and Y axis represents the height of the end-effector. From 0.0s to1.5s, R_1 followed the movement

of R_2 and lifted the stick to the height of 0.20 metres. This process was controlled by the first fuzzy controller. From 1.5*s* to 3.0*s*, R_1 tried to keep the tick level. This process is controlled by the second fuzzy controller. It can be seen from the diagram that there is an overshot between 1.8*s* and 3.0*s*. Then R_1 waited from 3.0*s* to 4.6*s*. From 4.6*s* to 5.2*s*, R_1 applied the test action of Lift_slightly, which is, lifting the stick for 0.05 metres and waited for R_2's response. In responding to the test action, at 5.2*s*, R_2 started to also lift the stick. Then the ARL system started to identify R_2's intention via its fuzzy backward reasoning. When the angle between the stick and ground was zero (corresponding to $\Delta Y = (0\ 1\ 0)$), the intention was identified via resolving the following fuzzy relational equation:

$$(a \quad b \quad c) \circ \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0.4 \\ 0 & 0 & 0.6 \end{pmatrix} = (0 \quad 1 \quad 0) \qquad (5.6)$$

Resolving the equation yielded a fuzzy set A' which is similar to but not A, which can be seen by comparing Figures 5.5(a) and 5.8. This deduced fuzzy set shows that R_2 did not change his mind, that is, he intended to have a rest and then continue. After R_2's intention was identified, the first fuzzy logic controller was switched on and R_1 continued to follow R_2 to lift the stick.

Figure 5.9 shows the angle between the stick and ground. *X* axis represents time and *Y* axis represents the angle. From 0.0*s* to 1.5*s*, R_1 followed R_2's movement. Because R_1 moved later than R_2, the angle increased to 0.10*rad*. From 1.5*s* to 3.0*s*, the angle decreased to -0.05*rad* because of the overshot. From 4.8*s* to 5.3*s*, the angle decreased to -0.03*rad* because R_1 applied the test action of Lift_slightly. From 5.3*s* to 6.2*s*, the angle increased to 0.06*rad* because R_2 continued to lift.

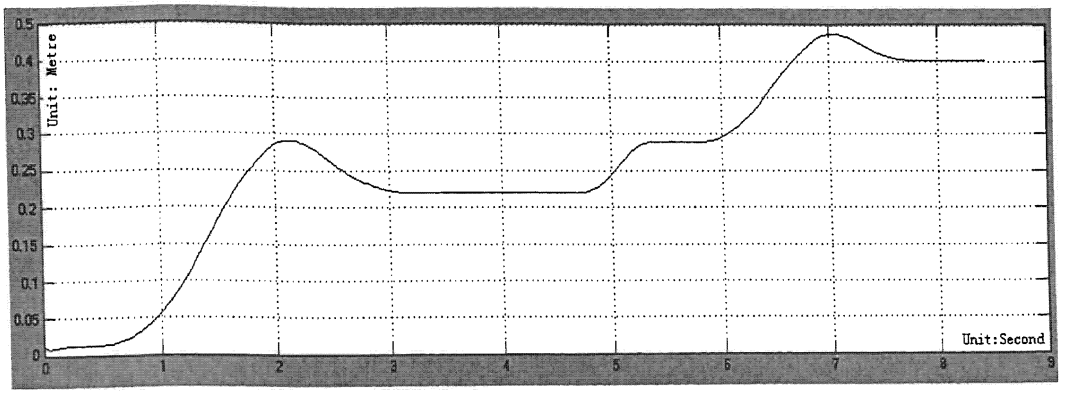

Figure 5.7 Trajectory of R_2's end-effector

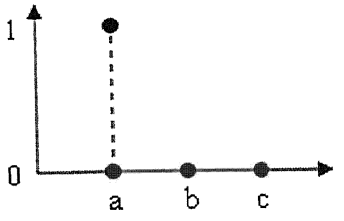

Figure 5.8 Fuzzy set A'

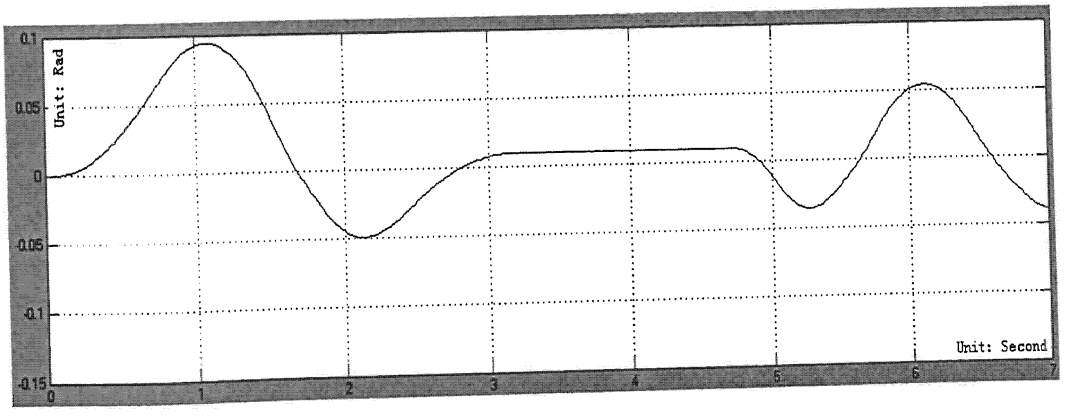

Figure 5.9 Angle between the stick and ground.

In the second simulation, the trajectory was designed for R_2 to lift the stick to a certain position and then put it down to simulate that R_2 changes mind. When having

detected $\Delta y = 0$, the ARL system in R_1 applied test action of Lift_slightly. R_2 at this time did not follow R_1 to lift the stick, instead, it started to put the stick down. When received this response, the ARL system started fuzzy backward reasoning and identified R_2's intention of "to put down". It then started to put the stick down.

The pre-defined motion trajectory for R_2 is shown in Figure 5.10. *X* axis represents time and *Y* axis represents the height of the end-effector. From 0.0*s* to1.5*s*, R_2 lifted the stick from the ground up to 0.23 metres. From 1.5*s* to 5.2*s*, R_2 stopped to have a rest. From 5.0*s* to 6.5*s*, R_2 put the stick down from the height of 0.23*m* to 0.03*m*. After 6.5*s*, R_2 stopped as the stick reached the ground.

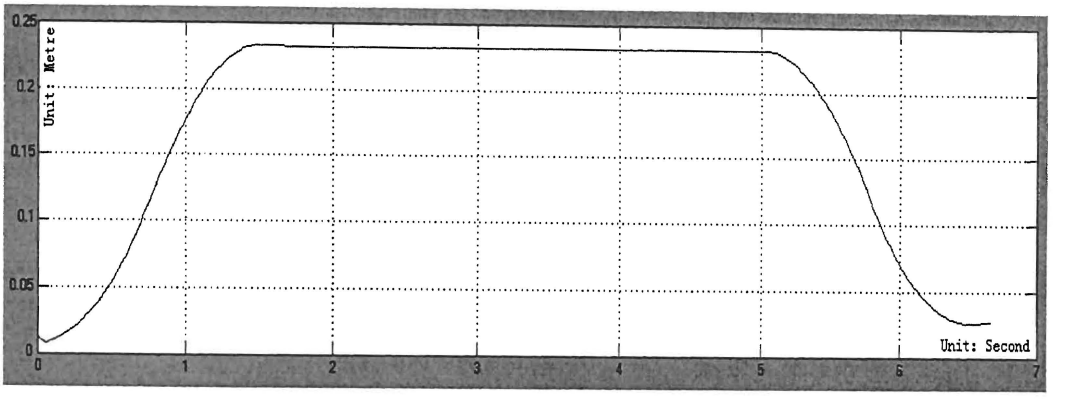

Figure 5.10 Trajectory of R_2's end effector

R_1's motion trajectory is shown in Figure 5.11. *X* axis represents time and *Y* axis represents the height of the end-effector. From 0.0*s* to1.5*s*, R_1 followed the movement of R_2 and lifted the stick to the height of 0.20 metres. This process was controlled by the first fuzzy controller. From 1.5*s* to 3.0*s*, R_1 tried to keep the tick level, controlled by the second fuzzy logic controller. It can be seen from the diagram that there is an overshot between 1.8*s* and 3.0*s*. Then R_1 waited from 3.0*s* to 4.6*s*. From 4.6*s* to 5.2*s*,

R_1 applied the test action of Lift_slightly, which is, lifting the stick for 0.05 metres and waited for R_2's response. At 5.2*s*, R_2 started to put the stick downward. The ARL system stated to identify R_2's intention. When the angle between the stick and ground was negative (corresponding to $\Delta Y = (0\ 0\ 1)$), the intention was identified through fuzzy backward reasoning via resolving the following fuzzy relational equation:

$$(a \quad b \quad c) \circ \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0.4 \\ 0 & 0 & 0.6 \end{pmatrix} = (0 \quad 0 \quad 1) \tag{5.7}$$

Resolving the equation yielded a fuzzy set C' which is similar to but not C, which can be seen by comparing Figures 5.5(c) and 5.12. This deduced fuzzy set shows that R_2 had changed his mind, that is, he intended to put the stick down to the ground. After R_2's intention was identified, R_1 also put the stick down and then followed R_2's movement.

The angle between the stick and ground is shown in Figure 5.13. *X* axis represents time and *Y* axis represents the angle. From 0.0*s* to 1.5*s*, R_1 followed R_2's movement. Because R_1 moved later than R_2, the angle increased to 0.10*rad*. From 1.5*s* to 3.0*s*, the angle decreased to -0.05*rad* because of the overshot. From 4.8*s* to 5.3*s*, the angle decreased to -0.07*rad* because R_1 applied the test action of Lift_slightly. From 5.3*s* to 6.2*s*, the angle continued to decrease to -0.10*rad* because R_2 put the stick down.

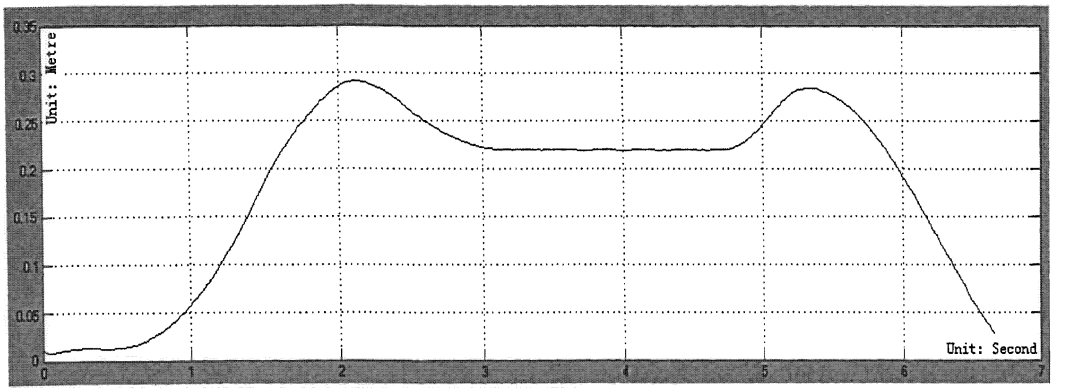

Figure 5.11 Trajectory of R_1's end-effector

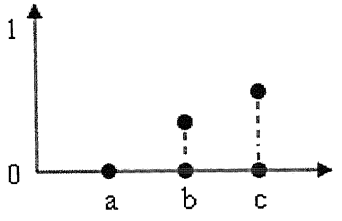

Figure 5.12 Fuzzy set *C'*

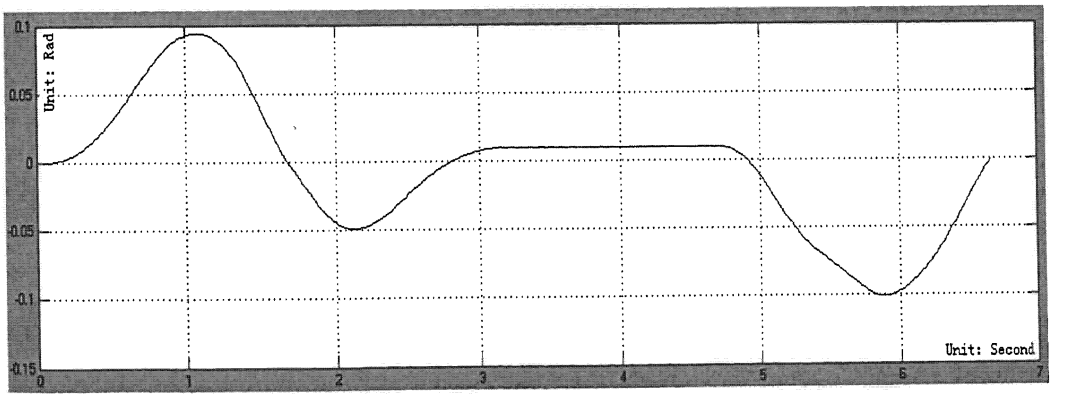

Figure 5.13 Angle between the stick and ground.

In the third simulation, the trajectory was designed for R_2 to lift the stick to a certain position and then stop for the purpose of simulating that the desired height is reached.

When having detected $\Delta y=0$, the ARL system in R_1 applied test action of Lift_slightly. R_2 in this time did not follow R_1 to lift the stick but stayed still. When having received this response, the ARL system started fuzzy backward reasoning and identified R_2's intention of "stay still". It then stopped.

The pre-defined motion trajectory for R_2 is shown in Figure 5.14. *X* axis represents time and *Y* axis represents the height of the end-effector. From 0.0*s* to1.5*s*, R_2 lifted the stick from the ground up to 0.23 metres. After 1.5*s*, R_2 stopped.

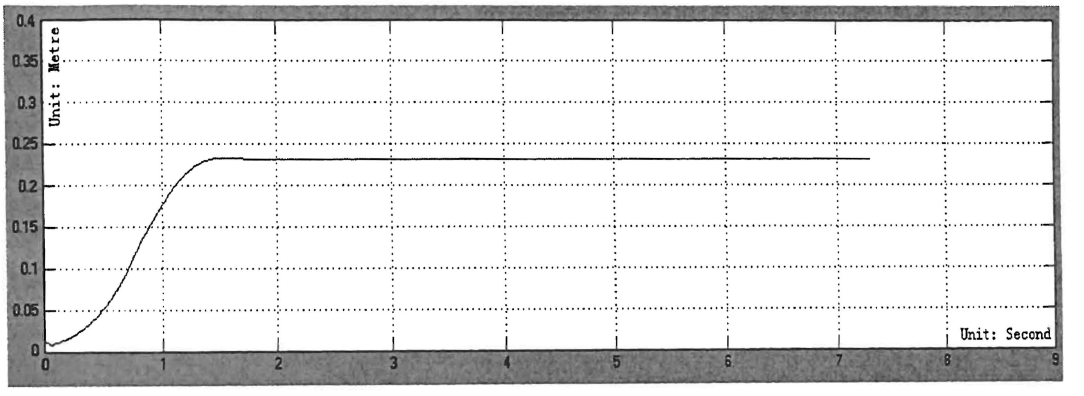

Figure 5.14 Trajectory of R_2's end-effector

R_1's motion trajectory is shown in Figure 5.15. *X* axis represents time and *Y* axis represents the height of the end-effector. From 0.0*s* to1.5*s*, R_1 followed the movement of R_2 and lifted the stick to the height of 0.20 metres. This process was controlled by the first fuzzy logic controller. From 1.5*s* to 3.0*s*, R_1 tried to keep the tick level controlled by the second fuzzy logic controller. It can be seen from the diagram that there is an overshot between 1.8*s* and 3.0*s*. Then R_1 waited from 3.0*s* to 4.6*s*. From 4.6*s* to 5.2*s*, R_1 applied the test action of Lift_slightly, which is, lifting the stick for 0.05 metres and waited for R_2's response. At 5.2*s*, the ARL system perceived R_2's

response -- the angle between the stick and ground was zero (corresponding to ΔY = (0 1 0)). It then started fuzzy backward reasoning to identify the intention via resolving the following fuzzy relational equation:

$$(a \quad b \quad c) \circ \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0.4 \\ 0 & 0 & 0.6 \end{pmatrix} = (0 \quad 1 \quad 0) \tag{5.8}$$

Resolving the equation yielded a fuzzy set B' which is similar to but not B, which can be seen by comparing Figures 5.5(b) and 5.16. This deduced fuzzy set shows that the desired height was reached. After R_2's intention was identified, R_1 started to put the stick to the original height, the desired height ($0.23m$) and kept the stick level.

The angle between the stick and ground is shown in Figure 5.17. X axis represents time and Y axis represents the angle. From $0.0s$ to $1.5s$, R_1 followed R_2's movement. Because R_1 moved later than R_2, the angle increased to $0.10rad$. From $1.5s$ to $3.0s$, the angle decreased to $-0.05rad$ because of the overshot. From $4.8s$ to $5.3s$, the angle decreased to $-0.05rad$ because R_1 applied the test action of Lift_slightly. From $5.3s$ to $6.0s$, the angle tended to zero.

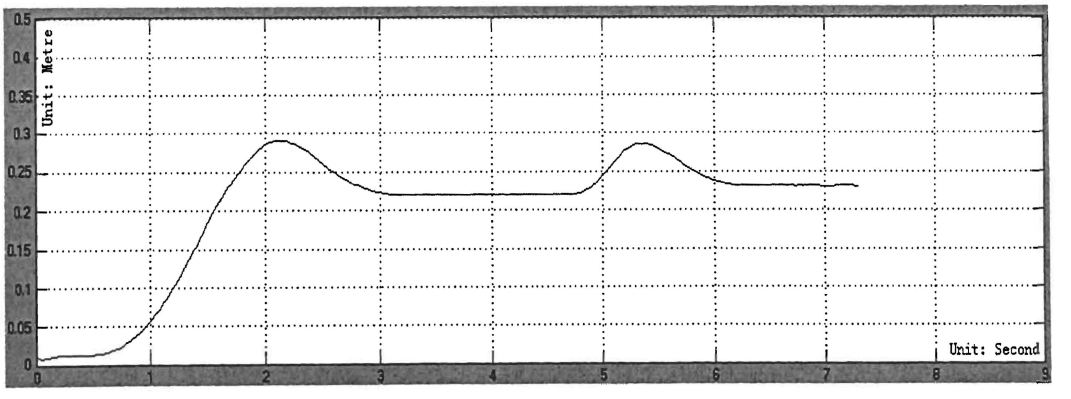

Figure 5.15 Trajectory of R_1's end-effector

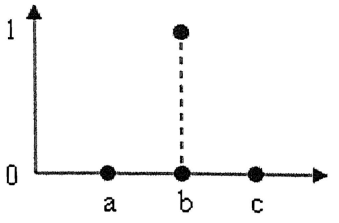

Figure 5.16 Fuzzy set B'

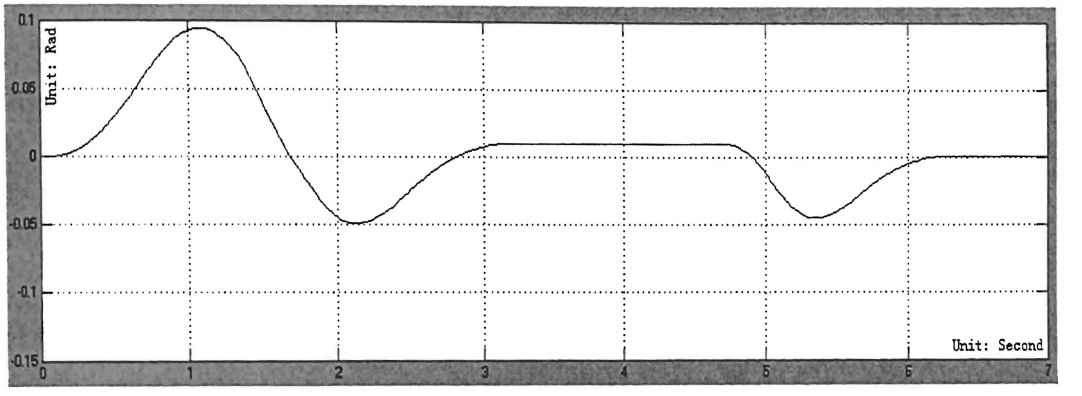

Figure 5.17 Angle between the stick and ground.

CHAPTER 6. CONCLUSIONS AND FURTHER WORK

6.1 Conclusions

This study aims at the development of the framework of ARL for cognitive robots, including the components and the relationship between the components. This framework allows a robot to actively recognize its user's intention/preference by testing the users and learning from the user's responses. This study also sets up a stick-lifting scenario to test the framework.

This study first carried out a survey of state-of-the-art approaches in the area of cognitive robots. Through the investigation, it has been found that the current robot learning approaches include the imitation based learning and award function based learning:

- The imitation based learning has two limitations. First, it only allows a robot to learn the user's intention passively. Second, it relies on social cues and therefore when using the robot the users must give exactly the same gestures as they act at the teaching stage to make sure the robot could pick up their intentions.
- The award function based learning also has limitations because the definition depends on how a robot's behaviour is parameterised. For different tasks, this function may have to be defined differently.

Secondly, this study has proposed the framework of active robot learning to facilitate a robot to develop the so-called high-order beliefs by actively collecting evidence it needs. The emphasis is on active learning, but not teaching. Hence social cues and award functions are not necessary. The following work has been done:

- Building a mathematical model of an industrial robot arm which was used in this study for the purpose of simulation. The robot arm has four links and can be used for object lifting. The links of the robot arm are driven by motors and the motors can be controlled by controllers.
- Building two fuzzy logic based controllers to control one robot arm to cooperate with another. One controller is used for robot cooperation and another is used for keeping the stick level.
- Developing a framework of active robot learning. This framework allows a robot to actively recognise its user's intention/preference.
- Setting up a stick-lifting scenario to test this framework. In this scenario, two robot arms have to fulfil a task of holding each end of a stick and lifting it to a certain height. The stick should be kept level during the whole process.
- Testing the fuzzy logic controllers and the framework of ARL through simulations.

The simulation results presented in this dissertation show:

- The fuzzy logic controllers can control a robot arm to cooperate with the other industrial robot arm effectively. When the industrial robot arm lifts one end of a stick, the one which is controlled by fuzzy logic controllers

can follow the movement of the industrial robot arm to lift the other end of the stick and keep the stick level.

- The ARL framework allows one robot arm to actively identify the intention/preference of the other one which was used to simulate a human.

6.2 Further Work

The fuzzy logic controller developed in this research for R_1 to track the movement of R_2 introduces a big overshot when R_2 stops to move. To overcome this overshot problem, some kind of damping, for example, the change rate of angle, may need to be introduced as one input of the controller. This signal can be obtained by comparing the current and the previous angle readings.

A flexible stick has been used in simulations to allow this study to focus on the high-order beliefs development. In reality, a stick can be a rigid body. To test if the proposed work is able to make a robot to cooperate with a human to pick up a rigid stick, force control is required for the robot.

In the process of user intention identification, the robot is able to actively testing the user. These test actions in this work have been developed according to the scenario of picking up a stick. For different scenarios, different kinds of test actions are required. Developing test actions according to scenarios may lead to a huge number of test actions and may confuse a robot when choosing a test action. More abstract test actions as well as arranging the test actions to ontology may need to be considered.

Accordingly, the inference engine that allows a robot to choose a suitable test action may also need to be reconstructed. Inspired by the process of interview, where interviewers may continuously question an interviewee before they have a satisfied answer, test actions which are related to a task may be chosen one after another before an intention is identified.

REFERENCES

Albus, J. S.: 4-D/RCS reference model architecture for unmanned ground vehicles, *IEEE International Conference on Robotics and Automation*, Vol. 3693, 24-27, April, 2000, San Francisco, CA, pp. 11-20

Arkin, R. C.: Behavior based robotics, *MIT Press*, May, 1998, Cambridge, MA.

Arnould, T. and Tano, S.: Interval-valued fuzzy backward reasoning, *IEEE Transactions on Fuzzy Systems,* Vol. 3, No.4, November, 1995, pp.425-437

Benjamin, D. P., Lonsdale, D. and Lyons, D.: A cognitive robotics approach to comprehending human language and behaviours, *Proceeding of the ACM/IEEE International Conference on Human-robot Interaction*, March, 2007, Arlington, Virginia, USA, pp. 185-192

Breazeal, C., Bauchsbaum, D., Gray, J., Gatenby, D. And Blumbery, B.: Learning from and about others: towards using imitations to bootstrap the social understanding of others, *Artificial Life*, Vol. 11, No.1-2, 2005, pp. 31-62

Brzostowski, J. and Kowalczyk, R.: Adaptive negotiation with on-line prediction of opponent behaviour in agent based negotiations, *2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'06)*, 18-22, December, 2006, Hong Kong, pp. 263-269

Burghart, C., Mikut, R., Stiefelhagen, R., Asfour, T., Holzapfel, H., Steinhaus, P. and Dillmann, R.: A cognitive architecture for a humanoid robot: a first approach, *IEEE-RAS International Conference on Humanoid Robots*, December 5-7, 2005, Tsukuba, Japan, pp. 357-362

Calinon, S. and Billard, A.: Teaching a humanoid robot to recognize and reproduce social cues, *The 15th IEEE International Symposium on Robot and Human Interactive Communication*, September, 2006, Hatfield, UK, pp. 346-351

Chella, A., Liotta, M. and Macaluso, I.: Cicerobot, a cognitive robot for museum tours, *An International Journal of Industrial Robot*, Vol. 34, No.6, 2007, Cambridge, CA, USA, pp. 503-511

Clement, B. J., Barrent, A. C. and Shaffer, S. R.: Argumentation for coordinating shared activities, *Proceedings of the 4th International NASA Workshop on Planning and Scheduling for Space*, 2004, CL#04-1368

Craig, J. J.: Introduction to Robotics Mechanics and Control, *Pearson Education*, 2005

Dearden, A. and Demiris, Y.: Learning forward models for robots, *Proceeding of IJCAI 2005*, March, 2005, Edinburgh, Scotland, pp.1440-1445.

Dempster, A. P.: A generalization of Bayesian inference. *Journal of the Royal Statistical Society*, Series B 30, 1968, pp. 205-247

Dillmann, R.: Teaching and learning of robot tasks via observation of human performance, *Robotics and Autonomous systems*, Vol. 47, No. 2-3, 2004, pp. 109-116

Duro, R. J., Becerra, J. A. and Santos, J.: Improving reusability of behaviour based robot cognitive architectures obtained through evolution, *Biologically Inspired Robot Behaviour Engineering*, 2003, pp. 239-259

Franke, J. and Otter, M.: The Manutec r3 benchmark models for the dynamic simulation of robots, *Technical Report*, TR R101-93, March, 1993

Hashimoto, H., Kubota, T., Kudou, M., Harashima, F.: Self-organizing visual servo system based on neural networks, *IEEE Control Systems Magazine*, Vol. 12, Issue 2, April, 1992, pp. 31-36

Hassch, A., Hohenner, S., Huwel, S., Kleinehagenbrock, M., Lang, S., Toptsis, I., Fink, G. A., Fritsch, J., Wrede, B. and Sagerer, G.: BIRON-The bielefeld robot companion, *Conference Paper of Workshop on Advances in Service Robotics*, May, 2004, Stuttgart, Germany, pp. 27-32

Kim, S., Song, Y., Kim, K., Cha, J. W. and Lee, G. G.: MMR based active machine learning for bio named entity recognition", *Proceedings of the Human Language*

Technology Conference of the North American, Chapter of the ACL, New York, June, 2006, pp.69-72

Leese, S. J.: Microphone arrays, In Davis, G. M., editor, Noise reduction in speech applications, *CRC Press*, 2002, Boca Raton, London, New York and Washington D. C., pp. 179-197

Li, D. and Zhang, J.: Collaborative agent system using fuzzy logic for optimisation, *Lecture Notes in Computer Science*, Vol. 2592, 2003, pp.195-210

Maes, P.: Modelling adaptive autonomous agents, *Artificial Life*, Vol. 1, Issue 1-2, 1994, Cambridge, MA, USA, pp. 135-162

Mamdani, E. H.: Application of fuzzy algorithms for control of simple dynamic plant, *Proceeding of IEE*, Vol. 121, No. 12, December, 1974, pp. 1585-1588

MATLAB Toolboxes, http://www.mathworks.com/

Petitt, J. D. and Braunl, T.: A framework for cognitive agents, *International Journal of Control, Automation, and Systems*, Vol. 1, No. 2, June, 2003, pp. 229-235

Pham, D. T. and Li, D.: Fuzzy control of a three-tank system, *Journal of Mechanical Engineering Science*, Part C, Vol. 215, No. C5, 2001

Robbel, P. and Vijayakumar, S.: Active learning for robot control, *Robotics Challenges for Processing Systems*, December, 2007, Vancouver, B.C., Canada

Sandini, G., Metta, G. and Vernon, D.: The iCub humanoid robot: an open-system research platform for enactive cognition, *50 Years of Artificial Intelligence*, 2006, pp. 358-369

Shafer, G.: A Mathematical Theory of Evidence, *Princeton University Press*, 1976

Tapus, A. and Mtaraic, M.: Towards active learning for socially assistive robots, *Robotics Challenges for Machine Learning Workshop, The 21st Annual Conference on Natural Information Proceeding Systems*, December, 2007, Vancouver, B.C., Canada

Vasudevan, S., Nguyen, V. and Siegwart, R.: Towards a cognitive probabilistic representation of space for mobile robots, *Proceeding of 2006 IEEE International Conference on Information Acquisition*, August, 2006, Shandong, China, pp. 353-359

Zadeh, L. A.: Fuzzy sets, *Information and Control 8*, 1965, pp. 338-353