This is to certify that the
dissertation entitled

Design and VLSI implementation of Perceptive Controller for Robotic Systems

presented by

Yu Sun

has been accepted towards fulfillment
of the requirements for the

Ph. D. degree in Electrical and Computer Engineering Department

Major Professor's Signature

9/22/04

Date

DESIGN AND VLSI IMPLEMENTATION OF PERCEPTIVE CONTROLLER FOR ROBOTIC SYSTEMS

By

Yu Sun

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering

2004

ABSTRACT

Design and VLSI implementation of Perceptive Controller for Robotic Systems

By

Yu Sun

Intuitively, human intelligence largely depends on the perception from interaction with the environment and the object. Similarly, robotic systems can sense the dynamic changes of the environment and the object states during the task execution. This dissertation is aimed at developing approaches for modeling, analyzing and designing robotic control systems based on the robotic perception, i.e., a perceptive frame.

In the perceptive frame, the model of a task can be learned by the robot based on the perception. In this dissertation, a model identification method for unknown parameters of a nonholonomic cart has been developed. Using interactions between a mobile manipulator and the cart, sensory information is accrued to estimate the model parameters of the cart. Since the raw data is contaminated by noise that cannot be modeled statistically, a wavelet based Least Square Method(LSM) is proposed to estimate these parameters for the cart. Experimental results verify that the estimation accuracy can be significantly improved by the use of the proposed lease square method.

For the control issues in the perceptive frame, an perceptive hybrid system approach for the motion planning and control of mobile robot systems is proposed. The ability to deal with unexpected events is one of the essential aspects of robot intelligence. Robotic systems can respond to environmental information obtained from sensors by making task decisions. According to the task decisions, the system is able to modify the original planned path and control the robot to execute the task autonomously. Perceptive model with single perceptive reference can deal with unexpected events only at the continuous level. It allows the perceptive reference to

be blocked and resumed by a class of unexpected events, including obstacles. The perceptive hybrid system model has three layers, one continuous layer, two discrete layers. Corresponding to the three layers, one continuous perceptive reference and two discrete perceptive references. The discrete layers enable the robot system to make decisions and modify original path plans. The properties of the system model, including linearity, continuity, stability in switching, and some dynamical properties of unexpected events have been discussed in the hybrid linguistic framework. The evolution of the hybrid perceptive references cannot be stopped by blocking the continuous perceptive reference. The properties have been verified by experiments.

For system implementation, a VLSI implementation oriented controller design for robotic system is proposed. For the robotic control systems in perceptive frame, the hybrid formal model is used for both the controller design and hardware description. The model can describe the high level behavior of the robotic system by a structural style architecture. The high level structure of the controller model guarantees the synthesizability of the control system. Therefore, the controller model can be mapped to VLSI implementation directly. A Cadence/Testbuilder system is used to simulate the hybrid controller design with a dynamics model. The HDL/C hybrid simulation results have verified that the hybrid system framework can be used for VLSI implementation, and the integrated system is synthesizable. It gives an efficient approach for hardware implementation of controllers for robotic systems.

to my family

ACKNOWLEDGEMENTS

Foremost, I would like to acknowledge all the invaluable help from my advisor, Dr. Ning Xi, without whom this would not have been possible. This dissertation comes from numerous discussions in his office, from his keen insight knowledge, from his guidance to a fruitful research area, and his perseverance and support. Not only the knowledge I learnt here, but also the research experience and friendship I gained here will be beneficial to the rest of my life.

I would like to thank all my committee members, Dr. Fathi Salam, Dr. Robert Schlueter and Dr. Baisheng Yan. In addition to their invaluable time they spend on the committee meeting, their insightful comments and suggestions will enhance the technical soundness of this dissertation.

I am grateful to my friends and colleagues from Robotics and Automation Lab. The discussions with Dr. Jindong Tan, Dr. Weihua Sheng, Dr. Jizhong Xiao, Amit Goradia, Dr. Imad Elhajj, Dr. Heping Chen, and Guangyong Li substantially contributed to my work and broaden my knowledge.

I dedicate this work to my family for their continuous emotional support: to my father and mother, for believing in me, to my sister-in-law, for everything she's done and finally to my brother, who set the standard for me, shared all my success and failures and stood by me through everything. Without their love and support, I would not have reached this point.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

The intelligence of robotic systems has been a longstanding focus of attention for control engineers. The robotic technology has increasingly wide application areas in a variety of environment, ranging from assembly tasks in manufacturing automation, underwater manipulations and home care services to space research. The environment in which the tasks are performed is varies widely. Furthermore, the environment and the task could be unstructured and dynamically changing. It requires the robotic systems to have human-like intelligence, i.e., process and respond to the sensed information relating to the task and environment.

Perceptive frame control is motivated by the demand for increased levels of intelligence of mobile robotic systems to deal with more complex tasks in an environment with increasing uncertainty. While the limited prior knowledge cannot provide the arguments rich enough to describe and execute the tasks. An obvious solution is to utilize perception on the control tasks and environment.

Perception can be expressed by events, both continuous and discrete, and composed by sensors. The human operator or planner, in this respect, thinks and plans in terms of discrete events. The traditional time-based control system design causes the inconvenience in improving the intelligence of the systems. As a low level trigger in the system, the time reference is an independent variable, it is not related to perception from sensors. Therefore, there are two naturally different types of references existing in the systems simultaneously, this fact may increases complexity in intelligence design and execution.

A unified perceptive reference frame is required for describing and utilizing the task or environmental perception. The objective of creating such a perceptive reference

is to build a triggering mechanism, expressing all the perceptive events, however, different from the time-based reference. Compared to the traditional time-based systems, the systems in perceptive frame can be driven by a perceptive reference, and described as the functions of perceptive reference, instead of the functions of time.

In task model learning, one of the challenges is the changing unstructured environment. The intelligent robotic systems have to face more complex tasks in a dynamical environment. The methods used for planning tasks for a static environment or assuming omniscient knowledge of the environment cannot be applied to dynamic environment. The specifications of real-time control systems require the control strategy to be able to "on-line" capture and process perceptive information.

The nature of the perceptive reference indicates that the interactions with the physical plant, through sensors and actuators, is analog, i.e. continuous. It also exhibits performance of the abstracted discrete references working at higher level to give the system more intelligence and flexibility. Discrete abstractions make it easier to manage the complexity of the system. Discrete models are easier to compute with, as all computers and algorithms are essentially discrete. In particular, It is worth noting that discrete abstractions make it easy to introduce the linguistic and qualitative information in the controller design.

The perceptive frame should be built to provide the information the control system need for characterizing the environment-control system interaction. It requires one to find a method to process hybrid references and task/action commands. The control methods in perceptive frame incorporate both continuous and discrete dynamics. Design and analysis of such a hybrid system is particularly challenging.

A survey of the ongoing research results in the context of the control methods in perceptive frame is given in the following section.

1.2 Literature Review

Task model learning, hybrid system modeling, and control and system design are the topics of this dissertation. Previous work related to these topics in the literature is reviewed.

Model Learning

In control system design, modeling the control objects is the foundation of high performance controllers. In most cases, the model is unknown before the task is performed. To learn the model, therefore, is essential. Generally, real-time systems use the sensors to measure the environment and the objects. The model learning task is particularly difficult when real-time control systems run in a noisy and changing environment, The measurements from the sensors may be contaminated by the non-stationary noise, i.e. it is changing randomly and depends on the environmental factors. The factors may include wind, vibration, friction and so on.

The most frequently used parameter identification methods are the Least Square Method (LMS) and the Kalman filter method Both have recursive algorithms for on-line estimation.

The Kalman filter [23] is used for parameter estimation. Generally, if a linear model or a linearized model of a dynamical system can be obtained, the system noise and observation noise are also known. The Kalman filter can estimate the states of the dynamic system through the observations regarding the system outputs. However, the estimation results of the Kalman filter are significantly affected by the system model and the noise models. Least Square Method can be applied to identify the static parameters in absence of accurate linear dynamic models.

There is rich literature on parameter estimation for robot manipulators. Zhuang and Roth [68] proposed a parameter identification method of robot manipulators. In this work, the Least Square Method is used to estimate the kinematic parameters based on a linear solution for the unknown kinematic parameters. To identify the

parameters in the dynamic model of the robot manipulator [35], a least square estimator is applied to identify the parameters of the linearized model. It is easy to see that LSM has been widely applied in model identification as well as in the field of robotics.

To achieve the objective of real-time online estimation, the Recursive Least Square Method(RLSM) has been developed to save computation resources and increase operation velocities for real time processing [30]. For identification, measurement noise is the most important problem. There are two basic approaches to processing a noisy signal. First, the noise can be described by its statistical properties, i.e. in time domain; second, a signal with noise can be analyzed by its frequency-domain properties.

Many algorithms of LSM are used to deal with noisy signals to improve estimation accuracy, but they require the knowledge of the additive noise signal. Durbin algorithm and Levinson-Wiener algorithm [22] require the noise to be a stationary signal with known auto-correlation coefficients. Each LMS based identification algorithm corresponds to a specific model of noise [31]. Iserman and Baur developed a Two Step Process Least Square Identification with correlation analysis [46]. But, for on-line estimation, especially in an unstructured environment, relation analysis results and statistical knowledge cannot be obtained. In this case, estimation results obtained by the traditional Least Square Method are very large(Table 2 in section 4). The properties of LSM in the frequency domain have also been studied. A spectral analysis algorithm based on least-square fitting was developed for fundamental frequency estimation in [14]. This algorithm operates by minimizing the square error of fitting a normal sinusoid to a harmonic signal segment. The result can be used only for fitting a signal by a monofrequent sinusoid.

Perceptive Frame Control

A continuous perceptive planning and control approach [60] has been applied to

the path planning and control for a single manipulator by Xi, Tarn and Bejczy. The basic idea of the perceptive planning and control theory is to introduce the concept of a motion reference s, a parameter that is directly relevant to the measured sensory outputs and the task. Instead of time, the control input is parameterized by the motion reference. Since the action reference is a function of the real time measurement, the values of the desired vehicle states are functions of the measured data. This creates a mechanism to adjust or modify the plan based on the measurements. Thus, the planning becomes a closed loop real-time process. The planner generates the desired values of the system, according to the on-line computed action reference parameter s. The perceptive approach guarantees the stability of the robot system in presence of unexpected events.

More results about the perceptive coordination of multiple manipulators were presented in [61]. Song, Tarn, and Xi [49][50][56][64][65] developed an perceptive scheme for the integration of task scheduling, action planning and control in robotic manufacturing systems, as shown in Figure 1.1. An approach by using Max-Plus Algebra model and perceptive planning and control scheme was proposed to provide an advanced mechanism to integrate manufacturing systems, so that task scheduling, which usually deals with continuous dynamics, can be treated in a unified analytical model. The interactions between discrete event systems and continuous dynamics are achieved so that the manufacturing systems are allowed to intelligently cope with unexpected system faults and uncertainties during the tasks. However the robot system cannot avoid the block from unexpected events, for instance, unexpected obstacles,

Hybrid System Modeling and Analysis

Hybrid system has been extensively investigated recently and a variety of hybrid system models and frameworks have been proposed[3][4][7][10][24][57][59].

Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho [2]

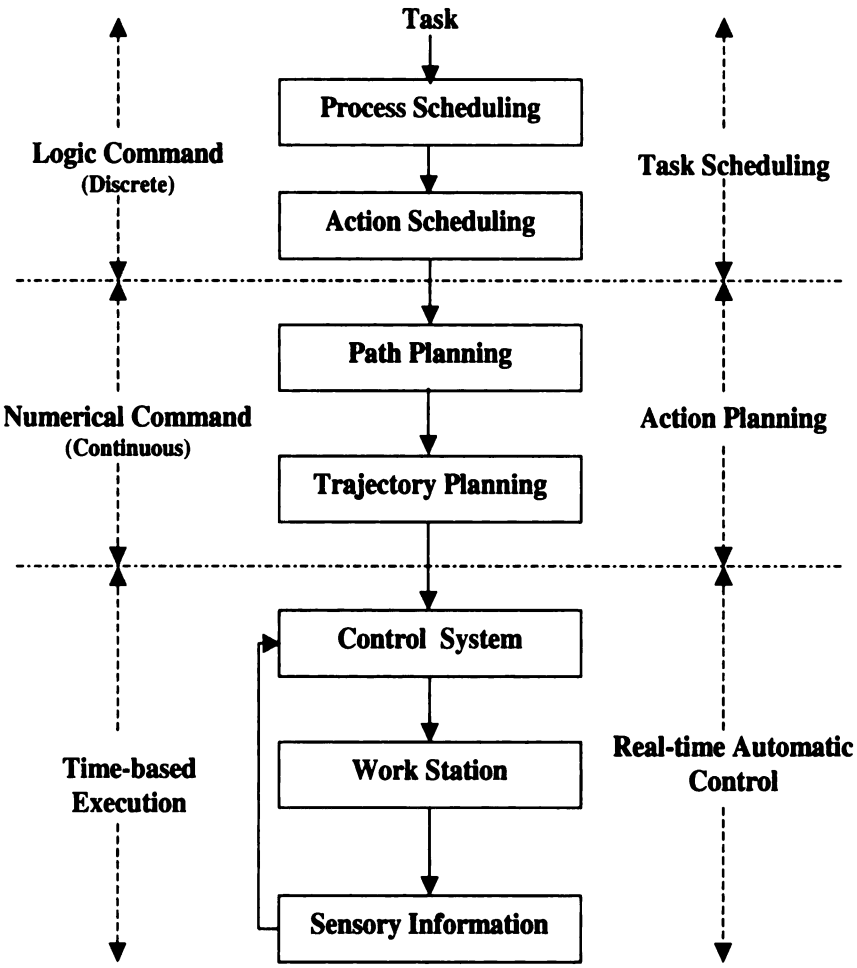


Figure 1.1: Hierarchical Scheduling, Planning and Control Scheme

proposed the concept of hybrid automata, as a model and specification language for hybrid systems. Hybrid automata can be viewed as a generalization of timed automata, in which the behavior of variables is governed in each state by a set of differential equations. The classification of the structures on hybrid automata based on a extended version can be found in [25] and [26].

For stability issues, multiple Lyapunov functions were introduced for hybrid and switched piecewise continuous systems. Branicky [5][6] proposed the multiple Lyapunov Function for switched and hybrid systems. Ye, Michel and Hou in [29][40][67] extended Branicky's result, and developed a general approach that can be applied to different types of systems that exhibit some kind of hybrid behavior.

In the research on hybrid automata's dynamical properties, the necessary and sufficient conditions for hybrid automata to have existence and uniqueness of execution were developed based on the nonblocking and the deterministic properties[38]. The dynamical properties, which include the continuity, invariance, and issues related to Lyapunov stability methods, were investigated in [36][37] by John Lygeros, Karl Henrik Johansson, Slobodan N. Simic, Jun Zhang and Shankar Sastry. As a hybrid system approach, hybrid automata theory has a serial of analysis and design methods. However, the switching conditions of the hybrid automata are fixed constraints, not flexible enough to describe the hybrid dynamics under the language-based input control.

Language and Linguistic Control

The application of languages, in this dissertation, is to solve the motion control problems in real-time systems. In the field of Artificial Intelligence (AI), the research on languages has been an attractive issue. AI is concerned with concepts and methods of symbolic inference by computer and symbolic knowledge representation for use in making inferences based on the use of languages. However, the sybolic languages only reflect actions and operations at the discrete levels of the intelligent systems [43]. In literature, Linguistic approaches are proposed to cope with discrete variables of hybrid systems and applied to the motion planning and control of mobile robots. The linguistic expressions can carry the control information for performing discrete and continuous operations on the physical systems.

The main problem in motion control is simultaneously achieving adequate speed and expressiveness. In order to solve the motion control problem, Brockett [8] defined the concept of an interpretive language and described a three-part solution to the problem of general motion control. It involves a motion description language together with an interpreter mechanism and an applications program. The language is, to a large extent, device independent and thus capable of describing the tasks in a wide

variety of systems. A more detailed construction criterion for the formal language is then present in [11].

Based on the idea of the interpretive language, Brockett [9] extends the concepts of the lattice language from the analysis of the robotic system motion. In [10], motion description language for kinetic state machines, which are the continuous analog of finite automata, is abstracted. The motion description language can support interactions between continuous and discrete aspects of a control system.

Furthermore, Manikonda at al. [39] defined atoms in the language alphabet that describes motion behaviors. The motion description language programs are composed by concatenating interrupt-driven "atoms". Transitions between atoms are triggered by changes in the environment or in the state of the robot. A library of atoms implements simple position and velocity-controlled movements as well as sensing operations involving the sonar, Infra-Red range finder and laser range finder. The functions of the language are improved to deal with multiple interruptions. In Egerstedt's research, by using motion description language of Brockett's hybrid model [16], the interaction between the continuous and the discrete parts can be given a meaningful control theoretic interpretation using reinforcement learning. An instruction complexity problem of avoiding multiple obstacles is analyzed. The results explain the way for choosing the motion alphabet such that the necessary instructions can be minimized when instructing a robot to navigate in cluttered environments populated by polygonal obstacles. By considering feedback in the finite automata, the instruction complexity of "free-running" automata was formulated in [16]. Free running finite automata can be stated as a finite state machine that reads an input and then makes transitions repeatedly, using the same input value, until a particular output condition is realized. The analysis shows that the length of the shortest description can be reduced by the feedback specification.

From the above review, it can be seen that Hybrid control schemes have been dis-

cussed exhaustively in the literature. However, most investigations of hybrid systems address the time-based descriptions, which is a special class of perception referenced systems. In other words, both the continuous part and the discrete part of the systems use time as the reference, little of them consider the hybrid system issues in perceptive frame. It is inconvenient, from the point of view of human intelligence. It is desirable to develop a perceptive control framework to unify the perception factors in hybrid system approach.

Because of the nature of the perception, the perceptive reference can be abstracted as symbolic inputs, linguistic expressions. The perceptive control approach, proposed in this dissertation, is inspired by the linguistic approaches discussed in [9][10][16].

We model the motion planning and control of the robotic systems by a hybrid system model based on perceptive motion reference. The perceptive motion reference hybrid model enables both the continuous part and the discrete part of the system to deal with unexpected events. This discrete part of the system "supervises" the robot system based on environmental sensor information such that the system can avoid unexpected obstacles by modifying original path plans. The discrete part and the continuous part are integrated by the perceptive motion reference.

Implementation-Oriented Perceptive Controller Design

Nowadays, there has been growing interest on implementation of digital systems with language description. For RTL level description, the typical implementation-oriented programming languages , for example, VHDL and Verilog have had an enormous impact on digital systems design methodology promoting a hierarchical top-down design process similar to the design of programs using high-level programming languages such as Pascal or C++ [47]. It has helped to establish new design methodology, taking the designers away from low level details such as transistors and logic gates to a much higher abstract level of the system description, just as high-level programming language take them away from the details of CPU. Unlike

other programming languages, VHDL and Verilog HDL provide the mechanisms to describe concurrent events being of crucial importance for the description of behavior of hardware.

The abstraction of the real time control system indicates that parallel execution is an important feature for system design. The high level programming language, for example, C++, Pascal, and JAVA can implement the parallel execution only with system calls [45]. The language does not evolve the parallel processing mechanism in its semantic description. Other than these programming languages, VHDL and Verilog take concurrency into consideration [44]. Its semantics enables the designers to synchronize the processes in a complex system at RTL level. However, there is a little literature about the VLSI implementation for real time control system design, especially, there is no such a method for direct hardware design of real time controller from the description at the behavioral level.

The real challenge in perceptive control is to develop a systemetic approach for design, analysis and implementation of hybrid robotic controller in a perceptive framework. The first technical difficulty is perception acquiring and processing, the perception is normally obtained from noisy sensory measurements. Signal processing and parameter identification will result in precise perceptive information for the purpose of control and planning. The second technical difficulty is perceptive control modeling and analyzing. The controller should be built on a perceptive framework, which provides the non-time based reference, and guarantees stability and other dynamical properties for dealing with unexpected events. The third difficulty is to create a mechanism to map the high level controller design in the perceptive frame to hardware, i.e., VLSI implementation without knowing the details of the circuit design.

1.3 Outline of the dissertation

The scope of this dissertation is to investigate new perceptive control methods for robot systems. The material of the dissertation is arranged in eight chapters.

Chapter 2 describes the mathematical tools for the perceptive signal and the perceptive frame model, Wavelet Transform and automata theory are introduced. Wavelet Transform theory is applied to non-stationary signals from a noisy environment. There is no prior information about the noise, i.e. containing non-modeled time-variant noise. The automata-formal language theory results in an efficient method to state the evolution behaviors of the discrete variables in physical systems.

Task model learning method in the perceptive frame is discussed in Chapter 3. In order to solve the model identification problem with unknown noise in an unknown environment, a wavelet based online model learning algorithm is developed. Using this algorithm, the characteristics of the unknown, non-stationary noise are ignored.

The objective of Chapter 4 is to develop a hybrid system approach for the planning and control of robotic manipulation in the perceptive frame. In this chapter, linguistic method is applied to system modeling. The robotic control systems are described in hybrid language-based perceptive frame.

The system analysis of the hybrid approach is given in Chapter 5. In the perceptive frame the evolving property, stability and complexity are considered with linguistic and hierarchical architecture.

Chapter 6 describes the experimental system, the experimental results for verifying both model learning and linguistic control approach are discussed.

Chapter 7 presents the discussion on the real-time system design and and VLSI implementation issues including synthesizability of the perceptive controller, hardware-software co-design and hybrid simulation for system verification.

Chapter 8 concludes this dissertation and suggests future work.

CHAPTER 2

PRELIMINARIES AND NOTATIONS

In this chapter, we cover the preliminaries and notations related to the model learning and the hybrid control method in the perceptive frame. The mathematical tools, including Wavelet Transform, automata theory, are introduced.

2.1 Wavelet Transform

The wavelet transform is a tool that cuts up data or functions or operators into different frequency components, and then studies each component with a resolution matched to its scale. This multiresolution concept is applied to wavelet transform. The wavelet transform has been successfully applied to non-stationary signals for analysis and has provided an alternative to the Windowed Fourier Transform.

The Fourier Transform, with its wide range of applications has its limitations. For example, this transformation cannot be applied to non-stationary signals that have time varying or spatial varying characteristics. Although the modified version of the Fourier Transform, referred to as Short-Time Fourier Transform(STFT) can resolve some of the problems associate with non-stationary signals, it does not address all the problems.

In this section, Discrete Wavelet Transform(DWT) is briefly introduced as the preliminary of the raw signal processing in on-line model identification.

2.1.1 Fourier Transform

The essence of the Fourier transform of a waveform is to decompose the waveform into a sum of sinusoids of different frequencies. In other words, the Fourier transform analyzes the "frequency contents" of a signal. The Fourier Transform identifies or distinguishes the different frequency sinusoids, and their respective amplitudes, which

combine to form an arbitrary waveform. The Fourier Transform is a frequency domain representation of a function.

2.1.2 Wavelet Transform and Short-Time Fourier Transform

The Short-Time Fourier Transform (STFT) replaces the Fourier transform's sinusoidal wave by the product of a sinusoid and a single analysis window which is localized in time. The STFT has a constant time frequency resolution. This resolution can be changed by rescaling the window. It is a complete, stable, redundant representation of the signal.

The STFT takes two arguments: time and frequency. It has a constant time frequency resolution. This resolution can be changed by rescaling the window.

Wavelet Transform is different from STFT.

The continuous wavelet transform is defined by

$$CWT_x^{\psi}(\tau, s) = \Psi_x^{\psi} = \frac{1}{\sqrt{(|s|)}} \int (x(t)\psi^*(\frac{t-\tau}{s}))dt \tag{2.1}$$

As seen in the above equation, the transformed signal is a function of two variables, τ and s , the translation and scale parameters, respectively. $\psi(t)$ is the transforming function, and it is called the mother wavelet . The term mother wavelet gets its name due to two important properties of the wavelet analysis as explained below:

The term wavelet means a small wave. The smallness refers to the condition that this (window) function is of finite length (compactly supported). The wave refers to the condition that this function is oscillatory. The term "mother" implies that the functions with different region of support that are used in the transformation process are derived from one main function, or the mother wavelet. In other words, the mother wavelet is a prototype for generating the other window functions.

The term "translation" is used in the same sense as it was used in the STFT; it is related to the location of the window, as the window is shifted through the signal. This term, obviously, corresponds to the time information in the transform domain. However, we do not have a frequency parameter, as we had before for the STFT. Instead, we have scale parameter which is defined as $(frequency)^{-1}$. The term frequency is reserved for the STFT.

2.1.3 Discrete Wavelet Transform

Wavelet Transform (WT) is a technique for analyzing signals. It was developed as an alternative to STFT to overcome problems related to its frequency and time resolution properties. More specifically, unlike the STFT that provides uniform time resolution for all frequencies the Discrete Wavelet Transform(DWT) provides high time resolution and low frequency resolution for high frequencies and high frequency resolution and low time resolution for low frequencies.

The DWT is a special case of the WT that provides a compact representation of a signal in time and frequency that can be computed efficiently. It can be explained as a subband coding.

The time-domain signal is passed from various highpass and low pass filters, which filters out either high frequency or low frequency portions of the signal. This procedure is repeated, every time some portion of the signal corresponding to some frequencies being removed from the signal.

The Continuous Wavelet Transform (CWT) is a correlation between a wavelet at different scales and the signal with the scale (or the frequency) being used as a measure of similarity. The continuous wavelet transform is computed by changing the scale of the analysis window, shifting the window in time, multiplying by the signal, and integrating over all times. In the discrete case, filters of different cutoff frequencies are used to analyze the signal at different scales. The signal is passed

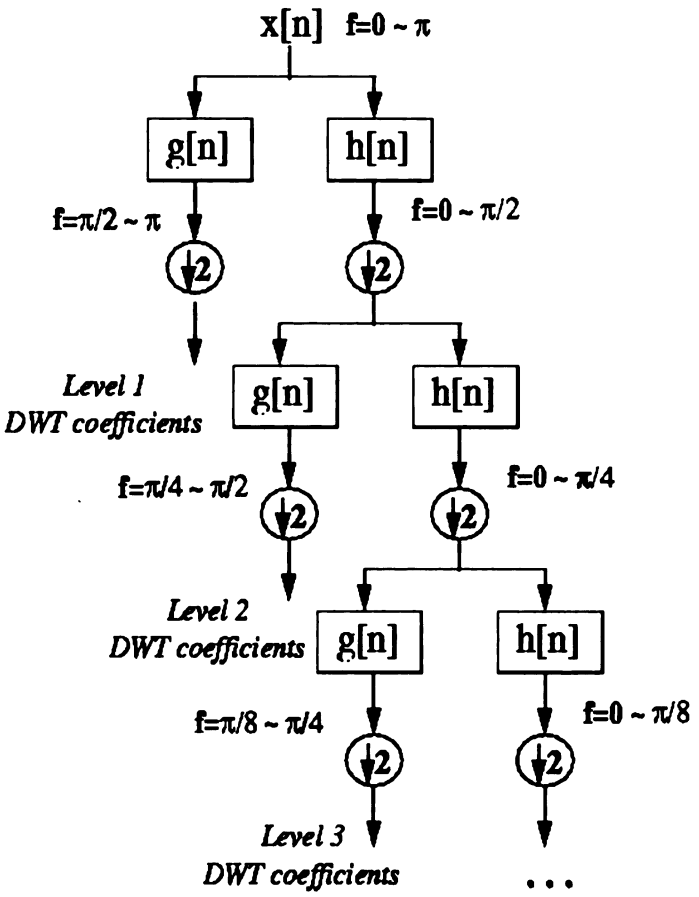


Figure 2.1: The multilevel Discrete Wavelet Transform.

through a series of high pass filters to analyze the high frequencies, and it is passed through a series of low pass filters to analyze the low frequencies. The subbanding procedure is shown in Figure 2.1.

In discrete signals, the Nyquist rate corresponds to π rad/s in the discrete frequency domain. After passing the signal through a half band lowpass filter, half of the samples can be eliminated according to the Nyquists rule, since the signal now has a highest frequency of $\pi/2$ radians instead of π radians. Simply discarding every other sample will subsample the signal by two, and the signal will then have half the number of points. The scale of the signal is now doubled. Note that the lowpass fil-

tering removes the high frequency information, but leaves the scale unchanged. Only the subsampling process changes the scale. Resolution, on the other hand, is related to the amount of information in the signal, and therefore, it is affected by the filtering operations. Half band lowpass filtering removes half of the frequencies, which can be interpreted as losing half of the information. Therefore, the resolution is halved after the filtering operation.

Next section will discuss Automata Theory used for perceptive control.

2.2 Automata Theory

Automata theory is the key technology in hybrid system modeling and system analysis. It provides a mathematical tool to describe the properties of the system in a discrete fashion. Finite Automata and formal languages are useful models for many important kinds of hardware and software. First we introduce some basic concepts about finite automata and formal languages.

A *symbol* is the abstract entity of automata theory. Examples are letters and digits.

An *alphabet* is a finite, nonempty set of symbols. Conventionally, we use the symbol Σ for an alphabet. Common alphabets include:

1. $\Sigma = 0, 1$, the binary alphabet.
2. $\Sigma = a, b, ..., z$, the set of all lower-case letters.
3. The set of all ASCII characters.

A *string* (or word) is a finite sequence of symbols chosen from some alphabet. The empty string, denoted ε, is the string consisting of zero symbols.

Concatenation of strings: Let x and y be strings. Then xy denotes the concatenation of x and y, that is, the string formed by making a copy of x and following it by a copy of y.

A set of strings all of which are chosen from some Σ^*, where Σ is a particular

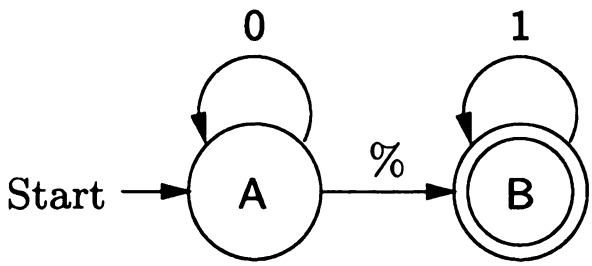


Figure 2.2: An Example of Finite Automata.

alphabet, is called a *language*.

If Σ is an alphabet, and $L \subseteq \Sigma^*$, then L is a language over Σ. Notice that a language over Σ need not include strings with all the symbols of Σ, so once L is a language over Σ, it is a language over any alphabet that is a superset of Σ.

A finite automaton(FA) is a system $(Q, \Sigma, \delta, q_0, M)$ where Q is a finite set of states, Σ is , , $q_0 \in Q$ is the initial state, $M \subset Q$ is the accepting states.

A finite automaton , M is a "tuple", $(Q, \Sigma, \Delta, q_0, F)$, where:

-Q is a finite set of states

-Σ is a set of input symbols, an alphabet, call the input alphabet

-$\Delta \subset Q \times \Sigma \rightarrow Q$ is a set of the transition mappings $Q \times \Sigma$ into Q.

-$F \subseteq Q$ is a set of final states.

Σ^* set of string of finite length of elements of Σ. Define string of length 0 to be ϵ.

M accepts a string $s \in \Sigma^*$, with $|s|$ = n, if there exists a sequence of states $q \in Q^*$, with $|q| = n + 1$ such that:

$-q[0] = q_0$ -For $i = 0, 1, ...n, (q[i], s[i], q[i+1]) \in \Delta$ $-q[n+1] \in F$.

The language accepted by M, L(M) is the set of all string s accepted by M.

Two automata, M1 and M2, are equivalent if L(M1)=L(M2).

An example of finite automata, automaton M is shown in Fig. 2.2. The state set consists of A and B . A is the starting state, B is the final state. The alphabet of the input symbols includes "$0, 1, and\%$". The arrows in the figure denoote the state transition. For automaton M, "0%" and "%1" are the words in the language

accepted by M.

2.3 Metric Space and Ordered Set

2.3.1 Metric Space

A metric space can be defined as a set S with a global distance function (the metric g) which, for every two points x, y in S, gives the distance between them as a nonnegative real number g(x,y). A metric space must also satisfy

1. $g(x, y) = 0$ if and only if $x = y$,
2. $g(x, y) = g(y, x)$,
3. The triangle inequality $g(x, y) + g(y, z) \geq g(x, z)$.

2.3.2 Ordered Set

A totally ordered set(or "linearly ordered set") is a set plus a relation on the set (called a total order) that satisfies the conditions for a partial order plus an additional condition known as the comparability condition. A relation is a total order on a set S ("$\leq$ totally orders S") if the following properties hold.

1. Reflexivity: $a \leq a$ for all $a \in S$.
2. Antisymmetry: $a \leq b$ and $b \leq a$ implies a = b.
3. Transitivity: $a \leq b$ and $b \leq c$ implies $a \leq c$.
4. Comparability (trichotomy law): For any $a, b \in S$, either $a \leq b$ or $b \leq a$.

The first three are the axioms of a partial order, while addition of the trichotomy law defines a total order. $\{S, \leq\}$ is called ordered space or totally ordered space, if S is an ordered set or a totally ordered set.

CHAPTER 3

TASK MODEL IN PERCEPTIVE FRAME

3.1 Introduction

A manipulator mounted on a mobile platform can significantly increase the workspace of the manipulation and its application flexibility. The applications of mobile manipulation range from manufacturing automation to search and rescue operations. A task such as a mobile manipulator pushing a passive nonholonomic cart can be commonly seen in manufacturing or other applications as shown in Fig.3.1.

The planning and control of the mobile manipulator and nonholonomic cart system is based on the mathematic model of the system. Some parameters of the model, including the mass of the cart and its kinematic length, are needed in the controller[54]. The kinematic length of a cart is defined on the horizontal plane as the distance between the axis of the front fixed wheels and the handle.

A nonholonomic cart can travel along its central line and perform turning movement about point C; in this case, the mobile manipulator applies a force and a torque on the handle at point A, as shown in Fig.3.2. The line between A and C is defined as the kinematic length of the cart, $|AC|$, while the cart makes an isolated turning movement. As a parameter, the kinematic length $|AC|$ of the cart can be identified by the linear velocity of point A and the angular velocity of line AC.

In cart-pushing, positions of the cart can be measured directly by multiple sensors. To obtain other kinematic parameters, such as linear and angular velocity of the object, numerical differentiation of the position data is used. This causes high frequency noises which are unknown in different environments. The unknown noise of the signal will cause large estimation errors. Experimental results have shown that

Figure 3.1: The mobile manipulator and a nonholonomic cart.

the estimation error can be as high as 90%. Therefore, the above Least square algorithms canot be used, and eliminating the effect of noise on model identification becomes essential.

This chapter presents a method for solving the problem of parameter identification for a nonholonomic cart modeling where the sensing signals are very noisy and the statistic model of the noise is unknown. When analyzing the properties of the raw signal in frequency domain, the noise signal and the true signal have quite different frequency spectra. In order to reduce the noise, a method to separate the noise signal and the true signal from the raw signal is used to process them in the frequency domain. A raw signal can be decomposed into several new signals with different bandwidths. These new signals are used to estimate the parameters; the best estimate is obtained by minimizing the estimation residual in the least square sense.

Based on this approach, combined with digital subbanding technique [1], a Wavelet Based Model Identification Method is proposed. The estimation convergence of the proposed model is proven theoretically and verified by experimentation. The experimental results show that the estimation accuracy is greatly improved without prior statistical knowledge of the noise.

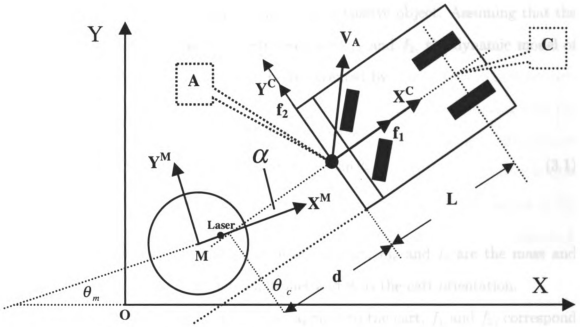


Figure 3.2: The associated coordinate frames of the mobile manipulator and a nonholonomic cart.

3.2 Interactive On-Line Model Learning

In this section, the task model learning problem is described.

3.2.1 Problem Formulation

A mobile manipulator usually consists of a mobile platform and a robot arm. Fig. 3.2 shows the coordinate frames associated with both the mobile platform and the manipulator. They include:

- **World Frame** Σ: XOY frame is Inertial Frame;
- **Mobile Frame** Σ_M: X^MMY^Mframe is a frame attached on the mobile manipulator;
- **Mobile Frame** Σ_A: X^CAY^Cframe is a frame attached on the nonholonomic cart.

The nonholonomic cart shown in Fig. 3.2 is a passive object. Assuming that the force applied to the cart can be decomposed into f_1 and f_2, the dynamic model of the nonholonomic cart in frame Σ can be represented by

$$\begin{aligned} \ddot{x}_c &= \frac{\lambda}{m_c}\sin\theta_c + \frac{f_1}{m_c}\cos\theta_c \\ \ddot{y}_c &= -\frac{\lambda}{m_c}\cos\theta_c + \frac{f_1}{m_c}\sin\theta_c \\ \ddot{\theta}_c &= \frac{f_2}{I_c} \end{aligned} \tag{3.1}$$

where x_c, y_c and θ_c are the configuration of the cart, m_c and I_c are the mass and inertia of the cart. λ is a Lagrange multiplier and θ_c is the cart orientation.

As shown in Fig. 3.2, The input forces applied to the cart, f_1 and f_2, correspond to the desired output force of the end-effector, f_x^d, f_y^d. In other words, the mobile manipulator controls the cart to achieve certain tasks by its output force. Therefore, manipulating the cart requires motion planning and control of the mobile manipulator based on the decoupled model [54], and the motion and force planning of the cart based on its dynamic model (3.1). An integrated task planning that combines both is desired.

The control input f_2 can then simply be designed [54] as

$$f_2 = \frac{I_c}{L}(\dot{\phi} - k_\theta(\dot{\theta}_c - \phi)) \tag{3.2}$$

Therefore the nonholonomic cart is largely characterized by L and the mass of the cart m_c. The objective of task model learning is to identify the two parameters.

The point A in Fig. 3.2 on the cart is the point on which the mobile manipulator force and torque are applied, C is the intersection between the rotation axis of the front wheels and the central line of the cart. V_A is the velocity of point A in frame Σ. θ_c, θ_m are the orientations of the cart and the mobile manipulator in frame Σ, respectively. α is the orientation of the cart in frame Σ_A .

It is known that L and m_c are two very important parameters in order to obtain the control input. While a mobile manipulator starts to manipulate an unknown cart. It is important to on-line identify L and m_c through the initial interaction with the cart. For simplification, the cart can be described as a segment of the central line of the cart. The line starts from its intersection with the rotation axis of front wheels (point C) and ended at the point(Point A) handled by the gripper. The length of the line is L. The turning movement of the object can be isolated to study its properties.

The geometric relationship between the angles is:

$$\theta_c = \alpha + \theta_m. \tag{3.3}$$

The on-board sensors in mobile manipulator include a Laser Range Finder, Encoders, and a Force/Torque Sensor. They can provide real-time sensory measurement of position of the mobile base and the end-effector, orientation of the cart, force and acceleration applied on the end-effector.

Based on the mobile manipulator sensor readings, the position of A can be described as:

$$\begin{bmatrix} x_a \\ y_a \end{bmatrix} = \begin{bmatrix} cos\theta_m & -sin\theta_m \\ sin\theta_m & cos\theta_m \end{bmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix} + \begin{bmatrix} x_m \\ y_m \end{bmatrix}. \tag{3.4}$$

The derivatives of the above equations are:

$$\dot{\theta_c} = \dot{\alpha} + \dot{\theta_m}, \tag{3.5}$$

$$\begin{bmatrix} \dot{x_a} \\ \dot{y_a} \end{bmatrix} = \begin{bmatrix} cos\theta_m & -sin\theta_m \\ sin\theta_m & cos\theta_m \end{bmatrix} \begin{bmatrix} \dot{p_x} \\ \dot{p_y} \end{bmatrix} + \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \end{bmatrix} + \dot{\theta_m} \begin{bmatrix} -sin\theta_m & -cos\theta_m \\ cos\theta_m & -sin\theta_m \end{bmatrix} \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \end{bmatrix}. \tag{3.6}$$

We define:

$$V_A = \begin{bmatrix} \dot{x_a} \\ \dot{y_a} \end{bmatrix}, \tag{3.7}$$

then $v^c_{ay} = \dot{x}_a \cdot sin\theta_c - \dot{y}_a \cdot cos\theta_c$.

According to the characteristics of a nonholonomic cart, the turning movement of the cart can be isolated as an independent motion. The movement can be described as

$$v^c_{ay} = L \cdot \dot{\theta}_c. \tag{3.8}$$

Equation (3.8) is the fundamental equation for identifying the parameter L.

To estimate the mass of the cart, the dynamics of the cart along X_c can be described by the following equation:

$$a_1 = \frac{f_1}{m_c} - \frac{F_f}{m_c}. \tag{3.9}$$

Where m is the mass of the cart, a_1 is acceleration on X_c axis, f_1 and F_f is the input force and the friction on X_c axis, respectively. a_1 and f_1 can be measured by the $Jr3$

force/torque sensor. Rewrite it in vector form

$$a_1 = \begin{bmatrix} f_1 \\ -1 \end{bmatrix}^T \cdot \begin{bmatrix} \frac{1}{m_c} \\ \frac{F_f}{m_c} \end{bmatrix}. \tag{3.10}$$

Based on (3.10), the mass of the cart can be identified by measurements of a_1 and f_1.

To estimate the kinematics length of a nonholonomic cart, positions, orientations, linear velocities and angular velocities of the cart are needed. The first two can be obtained by mobile manipulator sensor readings and previous work on orientation finding [54]. To find the velocity signals, a numerical differentiation method is used to obtain the derivative of position and orientation information. For a time domain signal, $Z(t)$, this procedure can be expressed by the following equation,

$$\dot{Z}(t) = \frac{Z(t) - Z(t-t')}{t-t'}, t - t' = \Delta t > 0. \tag{3.11}$$

Where Δt is the sampling time.

The orientation signal is very noisy due to the sensor measurement error. The noise involved in the orientation signal is significantly amplified after numerical differentiation, and it cannot be easily statistically modeled. The unmodeled noise causes a large estimation error in parameter identification.

3.2.2 State Estimation

Sensor Fusion: The mobile platform is equipped with a laser range finder. Figure 3.2 shows a configuration of the cart in the moving frame of the mobile platform. The laser sensor provides a polar range map of its environment in the form of (α, r), where α is a angle from $[-\pi/2, \pi/2]$ which is discretized into 360 units. The range sensor

provides a ranging resolution of $50mm$. Figure 3.3 (a) shows the raw data from the laser sensor. Obviously the measurement of the cart is mixed with the environment surround it. Only a chunk of the data is useful and the rest should be filtered out. A sliding window, $w = \{\alpha_1, \alpha_2\}$ is defined. The data for $\alpha \notin w$ are discarded. To obtain the size of w, the encoder information of the mobile manipulator should be fused with the laser data. Since the cart is hold by the end effector, the approximate position of x_r, y_r, as shown in Figure 3.2, can be obtained. Based on x_r, y_r and the covariance of the measurement r, which is known, the sliding window size can be estimated. The filtered data by the sliding window is shown in Figure 3.3(b).

In this applications, the cart position and orientation (x_c, y_c, θ_c) are the parameters to be estimated [55]. Assuming the relative motion between the cart the robot is slow, the cart position and orientation can be estimated by the standard Extended Kalman Filter(EKF) technique[32] or least square method. The cart dynamics are described by vector function:

$$\dot{X}_c = F(X_c, f) + \xi \tag{3.12}$$

where f is the control input of the cart, which can be measured by the force torque sensor equipped on the end-effector of the mobile manipulator.
$X_c = \{x_c, \dot{x}_c, y_c, \dot{y}_c, \theta_c, \dot{\theta}_c\}$. ξ is random vector with zero mean and covariance matrix σ_x. Considering the output vector as $Z(t) = \{\alpha, r\}^T$, as shown in Figure 3.2, the output function is described as:

$$Z(t) = G(X_c) + \zeta \tag{3.13}$$

where ζ is a random vector with zero mean and covariance matrix σ_z. It is worth noting that nonholonomic constraint should be considered in (3.12) and the measure-

ment (α, r) should be transformed into the moving frame. By applying the standard EKF technique to (3.12) and (3.13), an estimate of the cart state variables $\hat{X}_c$ for X_c can be obtained. The dotted line in Figure 3.3 (b) shows an estimation of the cart configuration in the mobile frame Σ_b. It is worthy noting that both the force/torque sensor information and the laser range finder information are used in the estimation of the cart configuration.

3.2.3 Wavelet Based Model Identification(WBMI)

The on-line parameter estimation problem can be described by a recursive Least Square Method.

Given a linear observation equation

$$Y_r = H_r X_r + v_r. \tag{3.14}$$

Y_r is an observation vector at time instance r, H_r is an observation matrix at time instance r. X_r is an estimated parameter vector. v_r is the measurement noise.

Corresponding to the estimation equations developed in section 2, for estimation of L, Y_r is the measurement sequence of V_{AY}^c, H_r is the measurement sequence of $\dot{\theta}_c$, X_r is the parameter L; for estimation of the mass, Y_r is the measurement sequence of the acceleration a_1, H_r is the sequence of the vector of $\begin{bmatrix} f_1 \\ -1 \end{bmatrix}^T$.

The parameter estimation task can be solved by Least Square Method[30].

The equations of the recursive least square method are:

$$\hat{X}_r = \hat{X}_{r-1} + K_r(Y_r - H_r'\hat{X}_{r-1}), \tag{3.15}$$

$$K_r = P_{r-1}H_r[1 + H_r'P_{r-1}H_r]^{-1}, \tag{3.16}$$

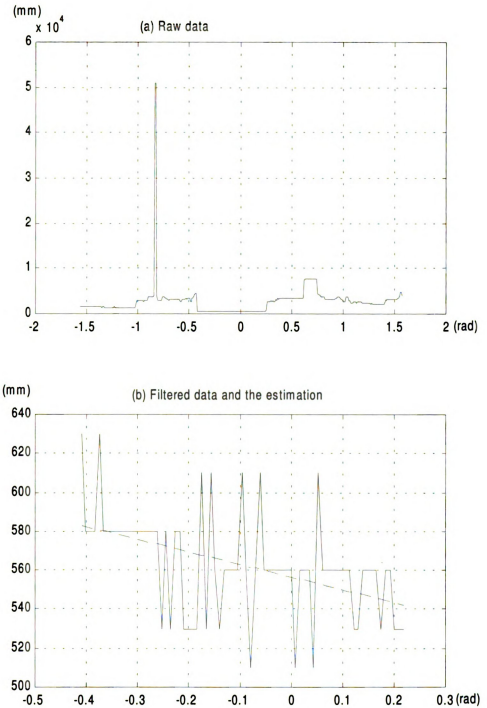


Figure 3.3: Laser range data

$$P_r = P_{r-1} - P_{r-1}K_r. \tag{3.17}$$

To obtain the best estimation of the kinematic length of the cart, we have to find the proper bandwidth, which causes the minimum estimation error. It is known that in the frequency domain the true signal is at low frequencies, the major parts of the noise are at high frequencies. Hence, the measured signal and the true signal have closer spectra at low frequencies than at high frequencies. Furthermore, to extract the proper low frequency part from measured signals will improve the estimation accuracy.

Daubechies Discrete Wavelet Transform(DWT) [15] is applied to decompose and reconstruct the raw signal in different bandwidth. In this dissertation, a raw signal will be sequentially passed through a series of Daubechies filters with imposed response $h_p(n)$ for wavelets with P vanishing moments.

To subband a signal, Discrete Wavelet Transform is used, each level of filtering can halve the bandwidth of the signal as shown in Fig 2.1.

As a result, the raw signal is preprocessed to have the desired low frequency components. The multiresolution approach from discrete wavelet analysis will be used to decompose the raw signal into several signals with different bandwidths. This algorithm makes the signal in this case, the raw angular velocity signal passes through several lowpass filters, in each level it passes the filter, the bandwidth of the signal would be halved, then the lower frequency component can be obtained level by level.

The algorithm can be described as the following procedures:

(a) Filtering: Passing the signal through a low pass Daubechies filter with bandwidth which is lower half bandwidth of the signal in the last level. Subsampling the signal by factor 2, then reconstructing the signal in this level;

(b) Estimating: Using the RLSM to process the linear velocity signal and the angular velocity signal obtained from the step (a) to estimate the kinematic

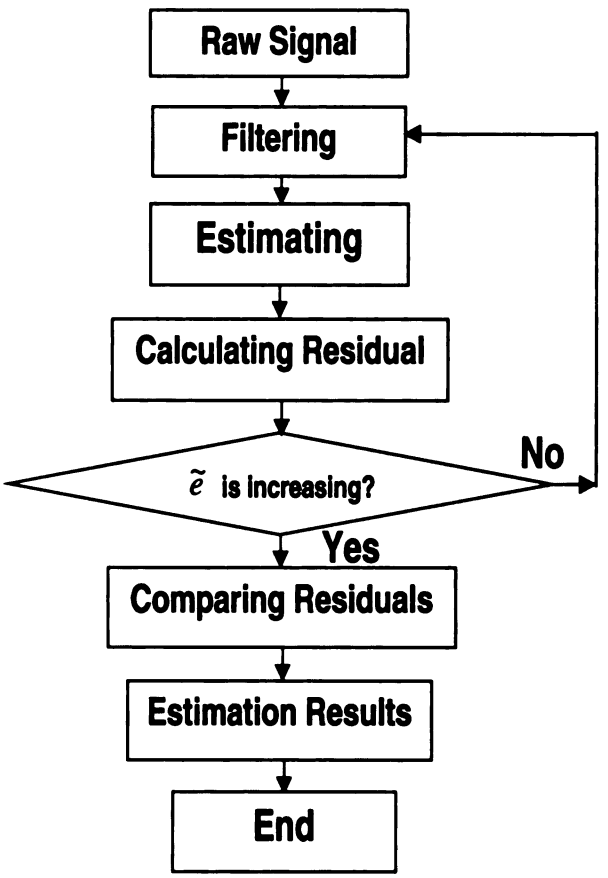


Figure 3.4: The block diagram of WBMI algorithm.

length of the cart.

(c) **Calculating**: Calculating the expectation of the length estimates and the residual.

(d) **Returning**: Returning to (a), until it can be ensured that the $\tilde{e}$ is increasing.

(e) **Comparing**: Comparing the residual in each level, take the estimate of length in a level, which has the minimum residual over all the levels, as the most accurate estimate.

The block diagram of the algorithm is shown in Figure 3.4.

3.3 Convergence of Estimation

The least square of estimation residual can be described by

$$\tilde{e} = \int_0^\tau [\hat{v}(t) - v(t)]^2 dt \tag{3.18}$$

and other relationships can be defined as follows:

$$v(t) = L \cdot \omega_T(t), \tag{3.19}$$

$$\hat{v}(t) = \hat{L} \cdot \omega_M(t), \tag{3.20}$$

$$\hat{L} = \Delta L + L, \tag{3.21}$$

$$\omega_M(t) = \omega_T(t) + \Delta\omega(t), \tag{3.22}$$

$$F_{\omega_M}(\omega) = F_{\omega_T}(\omega) + \Delta F(\omega). \tag{3.23}$$

L is the true value of the length, $\hat{L}$ is the estimate of the length in least square sense. $v(t)$ is the true value of the linear velocity, $\hat{v}(t)$ is the estimate of the linear

velocity, $\omega_T(t)$ is the true value of $\dot{\theta}_c$, $\omega_M(t)$ is the measurements of $\dot{\theta}_c$ and $\Delta\omega(t)$ are measurement additive noise signal of $\dot{\theta}_c$, respectively. $F_{\omega_T}(\omega)$, $\Delta F(\omega)$ and $F_{\omega_M}(\omega)$ are their corresponding Fourier Transforms.

Considering the problem as a minimizing problem, the estimation error can be minimized by finding the minimum value of the estimation residual $\tilde{e}$ in least square sense. The estimation residual is in terms of the frequency domain form $\Delta F(\omega)$ of the error signal $\Delta\omega(t)$. Hence, the problem is turned into describing the relation between the $\tilde{e}$ and $\Delta F(\omega)$.

The following lemma provides a conclusion that functions with a certain form are increasing functions of a variable. Based on the lemma, a theorem can be developed to prove that $\tilde{e}$ is a function of $(\frac{\Delta L}{L})^2$ which has the same form as in the lemma. Thus, the estimation error decreases, as the residual decreases.

Lemma : Let $\Omega : (-\infty, +\infty)$ and

$$X = \left(\frac{\int_\Omega F_{\omega_M}(\omega)\overline{\Delta F(\omega)}d\omega}{\int_\Omega |F_{\omega_M}(\omega)|^2 d\omega}\right)^2, \tag{3.24}$$

$$\tilde{e} = \frac{L^2}{2\pi}\left(\int_\Omega |\Delta F(\omega)|^2 d\omega - \int_\Omega |F_{\omega_M}(\omega)|^2 d\omega \cdot X\right). \tag{3.25}$$

If $F_{\omega_T}(\omega)$ is orthogonal to $\Delta F(\omega)$, then $\tilde{e}$ is a strictly increasing function of X.

Proof : $\Delta F(\omega)$ is orthogonal to $F_{\omega_T}(\omega)$, i.e.

$$\int_\Omega F_{\omega_T}(\omega)\overline{\Delta F(\omega)}d\omega = 0. \tag{3.26}$$

Simplifying the integrals

$$\begin{aligned}\int_\Omega F_{\omega_M}(\omega)\overline{\Delta F(\omega)}d\omega &= \int_\Omega (F_{\omega_T}(\omega)+\Delta F(\omega))\overline{\Delta F(\omega)}d\omega = \int_\Omega |\Delta F(\omega)|^2 d\omega,\\ \int_\Omega |F_{\omega_M}(\omega)|^2 d\omega &= \int_\Omega |F_{\omega_T}(\omega)|^2 d\omega + \int_\Omega |\Delta F(\omega)|^2 d\omega.\end{aligned}$$

can move out some terms in X, it is clear that X is a real function as

$$X = \left(\frac{\int_\Omega |\Delta F(\omega)|^2 d\omega}{\int_\Omega (|F_{\omega_T}(\omega)|^2 + |\Delta F(\omega)|^2) d\omega}\right)^2.$$

It implies

$$\int_\Omega |\Delta F(\omega)|^2 d\omega = \frac{\sqrt{X}}{1-\sqrt{X}} \int_\Omega |F_{\omega_T}(\omega)|^2 d\omega. \tag{3.27}$$

$\tilde{e}$ can be expressed in terms of X

$$\begin{aligned}\tilde{e} &= \frac{L^2}{2\pi}\left(\int_\Omega |\Delta F(\omega)|^2 d\omega - \int_\Omega |F_{\omega_M}(\omega)|^2 d\omega \cdot X\right)\\ &= \frac{L^2}{2\pi}\left(\int_\Omega |\Delta F(\omega)|^2 d\omega - X\cdot\int_\Omega |F_{\omega_T}(\omega)|^2 d\omega - X\cdot\int_\Omega |\Delta F(\omega)|^2 d\omega\right)\\ &= \frac{L^2}{2\pi}(1-\sqrt{X})\frac{\sqrt{X}}{1-\sqrt{X}}\cdot\int_\Omega |F_{\omega_T}(\omega)|^2 d\omega\\ &= \frac{L^2\int_\Omega |F_{\omega_T}(\omega)|^2 d\omega}{2\pi}\sqrt{X}.\end{aligned}$$

Let $f(X) = \tilde{e}$, then

$$f'(X) = \frac{L^2\int_\Omega |F_{\omega_T}(\omega)|^2 d\omega}{4\pi\sqrt{X}} > 0. \tag{3.28}$$

Hence, given $\int_\Omega |F_{\omega_T}(\omega)|^2 d\omega$, $f(X)$ is an increasing function of X.

Finally, $\tilde{e}$ is an increasing function of X. If $\tilde{e} = 0$, $X = 0$.

The lemma provides a foundation to prove $(\frac{\Delta L}{L})^2$ will reach a minimum value

when the estimation residual $\tilde{e}$ takes a minimum value.

Theorem : Given $\Delta F : \omega \rightarrow C$, C is a complex space. When $\tilde{e}$ takes a minimum value, $(\frac{\Delta L}{L})^2$ also takes a minimum value.

Proof : Consider the continuous case:

$$\tilde{e} = \int_0^\tau [\hat{v}(t) - v(t)]^2 dt = \int_0^\tau [\hat{v}(t)^2 - 2\hat{v}(t)v(t) + v(t)^2] dt.$$

Given $\Omega : (-\infty, +\infty)$, according to Parserval's Equation [41],

$$\tilde{e} = \frac{1}{2\pi} \int_\Omega (|F_{\hat{v}}(\omega)|^2 - 2F_{\hat{v}}(\omega)\overline{F_v(\omega)} + |F_v(\omega)|^2) d\omega.$$

From (3.20) and linear properties of Fourier Transform, it can be easily seen that

$$\tilde{e} = \frac{1}{2\pi} \int_\Omega (\hat{L}^2 |F_{\omega_M}(\omega)|^2 - 2\hat{L}F_{\omega_M}(\omega)\overline{F_{\omega_T}(\omega)}) d\omega + \frac{1}{2\pi} \int_\Omega (|F_v(\omega)|^2 d\omega. \qquad (3.29)$$

$\tilde{e}$ is a function of $\hat{L}$, then based on the least square criterion the following equation in terms of $\hat{L}$ must satisfy

$$\frac{\partial \tilde{e}}{\partial \hat{L}} = \frac{2\hat{L}}{2\pi} \int_\Omega (\hat{L}^2 |F_{\omega_M}(\omega)|^2 - \hat{L}F_{\omega_M}(\omega)\overline{F_v(\omega)}) d\omega = 0.$$

The above equation implies that the solution of $\hat{L}$ can be expressed as

$$\int_\Omega (\hat{L}|F_{\omega_M}(\omega)|^2 - F_{\omega_M}(\omega)\overline{F_v(\omega)}) d\omega = 0.$$

Using (3.19) yields

$$\hat{L} = \frac{\int_\Omega F_{\omega_M} \overline{F_v(\omega)} d\omega}{\int_\Omega |F_{\omega_M}(\omega)|^2 d\omega} = \frac{L \int_\Omega F_{\omega_M} \overline{F_{\omega_T}(\omega)} d\omega}{\int_\Omega |F_{\omega_M}(\omega)|^2 d\omega} \tag{3.30}$$

Let $L^* = \hat{L}$, substituting (3.21),(3.23)into (3.30) for removing the terms of the linear velocity yields

$$\frac{L + \Delta L}{L} = \frac{\int_\Omega (|F_{\omega_M}(\omega)|^2 - F_{\omega_M}(\omega)\overline{\Delta F(\omega)}) d\omega}{\int_\Omega |F_{\omega_M}(\omega)|^2 d\omega}. \tag{3.31}$$

There exists the relation between the estimation error ($\frac{\Delta L}{L}$) in the time domain and the measurement error in the frequency domain $\Delta F(\omega)$,

$$\frac{\Delta L}{L} = -\frac{\int_\Omega F_{\omega_M}(\omega)\overline{\Delta F(\omega)} d\omega}{\int_\Omega |F_{\omega_M}(\omega)|^2 d\omega}. \tag{3.32}$$

Note X is defined in the beginning of the section, then $X = (\frac{\Delta L}{L})^2$

Substituting (3.30) into (3.29) yields

$$\tilde{e} = \frac{L^2}{2\pi}(-\frac{\int_\Omega F_{\omega_M} \overline{\Delta F(\omega)} d\omega}{\int_\Omega |F_{\omega_M}(\omega)|^2 d\omega} + \int_\Omega (|F_{\omega_T}(\omega)|^2 - |F_{\omega_M}(\omega)|^2 + 2F_{\omega_M}(\omega)\Delta F(\omega)) d\omega) \tag{3.33}$$

We define:

$$\Delta e = \int_0^\tau (\Delta\omega(t))^2 dt = \int_0^\tau [\omega_M(t)^2 - 2\omega_M(t)\omega_T(t) + \omega_T(t)^2] dt.$$

Applying Parserval's Equation to the error signal $\Delta\omega$ yields

$$\int_\Omega |\Delta F(\omega)|^2 d\omega = \int_\Omega (|F_{\omega_T}(\omega)|^2 + |F_{\omega_M}(\omega)|^2 - 2F_{\omega_M}(\omega)\overline{F_{\omega_T}(\omega)}) d\omega.$$

$$= \int_{\Omega}(|F_{\omega_T}(\omega)|^2 d\omega + \int_{\Omega}|F_{\omega_M}(\omega)|^2 d\omega - 2\int_{\Omega} F_{\omega_M}(\omega)(\overline{F_{\omega_M}(\omega) - \Delta F(\omega)})d\omega.$$

Therefore,

$$\int_{\Omega}(|F_{\omega_T}(\omega)|^2 - |F_{\omega_M}(\omega)|^2)d\omega = \int_{\Omega}(|\Delta F(\omega)|^2 - 2F_{\omega_M}(\omega)\overline{\Delta F(\omega)})d\omega. \tag{3.34}$$

Substituting (3.7),(3.16) into (3.15) $\tilde{e}$ can be given in terms of X

$$\tilde{e} = \frac{L^2}{2\pi}(\int_{\Omega}|\Delta F(\omega)|^2 d\omega - X \cdot \int_{\Omega}|F_{\omega_M}(\omega)|^2 d\omega). \tag{3.35}$$

It can be easily seen that $\tilde{e}$ has the same form as in the lemma, then $\tilde{e}$ is an increasing function of X, for different ΔF, when $\tilde{e}$ takes a minimum value, $(\frac{\Delta L}{L})^2$ also takes a minimun value. Since the minimum value of $\tilde{e}$ is equal to 0, the $(\frac{\Delta L}{L})^2$ will approach 0 as well. The residual of the estimation is convergence and the estimation error goes to 0.

CHAPTER 4
LINGUISTIC APPROACH FOR ROBOTIC SYSTEM CONTROL IN PERCEPTIVE FRAME

4.1 Introduction

In recent years, there has been increasing interest in robot intelligence. As applications are becoming more and more complicated, robot systems need to have a certain amount of autonomy. One of the most important aspects of autonomous systems is the ability to deal with unexpected events in a changing environment. The robot processes environmental information obtained from onboard sensors, and responds to the information by changing the original path planning and control schemes. Based on the perceptive-based approach [60], a hybrid perceptive reference model is proposed, which enables the mobile robot system to autonomously respond to unexpected events.

A continuous perceptive planning and control approach [60] has been applied to the path planning and control for a single manipulator, and multiple manipulators coordination [61]. Song et al. [49] developed an perceptive scheme for integration of task scheduling, action planning and control in robotic manufacturing systems. The basic idea of the perceptive planning and control theory is to introduce the concept of a motion reference s, a parameter that is directly relevant to measured sensory outputs and the task. Instead of time, the control input is parameterized by the motion reference. since the action reference is a function of the real time measurement, the values of the desired vehicle states are functions of the measured data. This creates a mechanism to adjust or modify the plan based on the measurements. Thus, the planning becomes a closed loop real-time process. The planner generates the desired

values of the system, according to the on-line computed action reference parameter s. Figure 4.1 indicates the perceptive control in comparison with the traditional control concepts.

The perceptive frame approach guarantees stability of the robot system in presence of unexpected events. However the robot system cannot avoid the obstacle based on the environmental sensor information. The hybrid perceptive reference model has three layers, one continuous layer, and two discrete layers. The continuous layer guarantees the system stability in the presence of unexpected events based on the perceptive approach. The discrete layers enable the robot system to make decisions and modify original path plans.

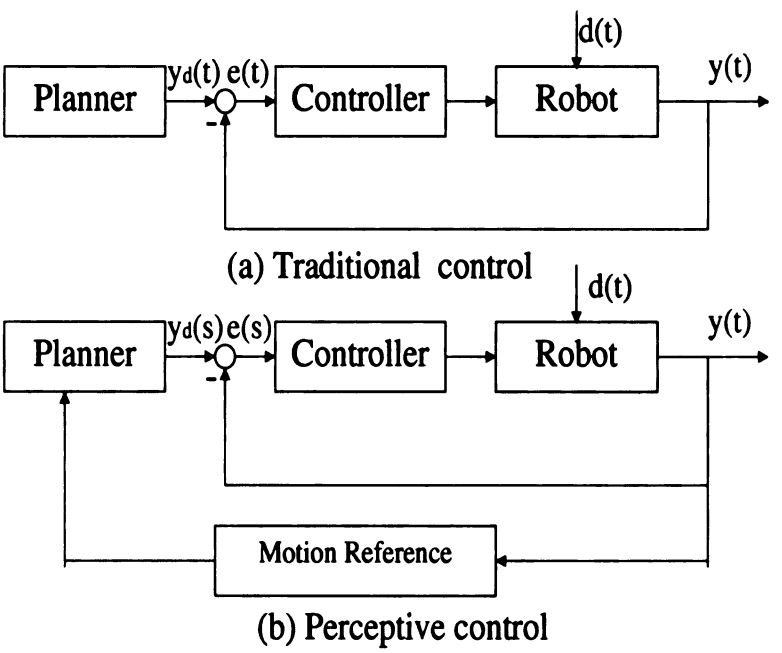


Figure 4.1: Perceptive Control

Linguistic/automata approach is the method applying the mathematical tools, formal languages and automata theory, to describe the dynamics of the control system at discrete levels. Hybrid control system behaviors, thus, can be discussed by hybrid automata, exhibiting both continuous and discrete evolutions. Brockett abstracted the concepts of the lattice language based on the analysis of the robot motion[9].

A motion description language for kinetic state machines, which are the continuous analog of finite automata, is proposed in [10]. Furthermore, Manikonda at al. [39] defined atoms in the language alphabet that describes motion behaviors. The functions of the language are improved to deal with multiple interruptions. In Egerstedt's research, a complexity problem of a multiple obstacle avoidance is analyzed by using motion description language of Brockett's hybrid model [16]. However, both the continuous part and the discrete part of the system use time as the reference. In this chapter, we model the motion planning and control of the mobile robot system by a hybrid system model based on perceptive motion reference. The perceptive motion reference hybrid model enables both the continuous part and the discrete part of the system deal with unexpected events. This discrete part of the system "supervise" the robot system based on environmental sensor information such that the system can avoid unexpected obstacles by modifying original path plans. The discrete part and the continuous part are integrated by the perceptive motion reference.

4.2 Hybrid Perceptive Reference Model for Control of Robotic System

A hybrid automaton based model, as shown in Figure 4.2, is developed for robot system to process both continuous and discrete information, and to give the discrete levels more intelligence. The system is composed of two major parts, a hierarchical command execution chain including task scheduler, action planner and motion planner, a hybrid perceptive reference chain including task level reference, action level reference and motion reference. The task scheduler, action planner, motion planner, task level reference and action level reference are modeled by hybrid languages and/or hybrid automata. The motion reference s is a continuous variable which is computed based on the configuration sensors on the robot system. The concept of hybrid per-

ceptive reference is proposed to model a top-down decision making system under the circumstance of unexpected events. The decision making mechanisms involved in this model can effectively use the sensory information for higher level decision making.

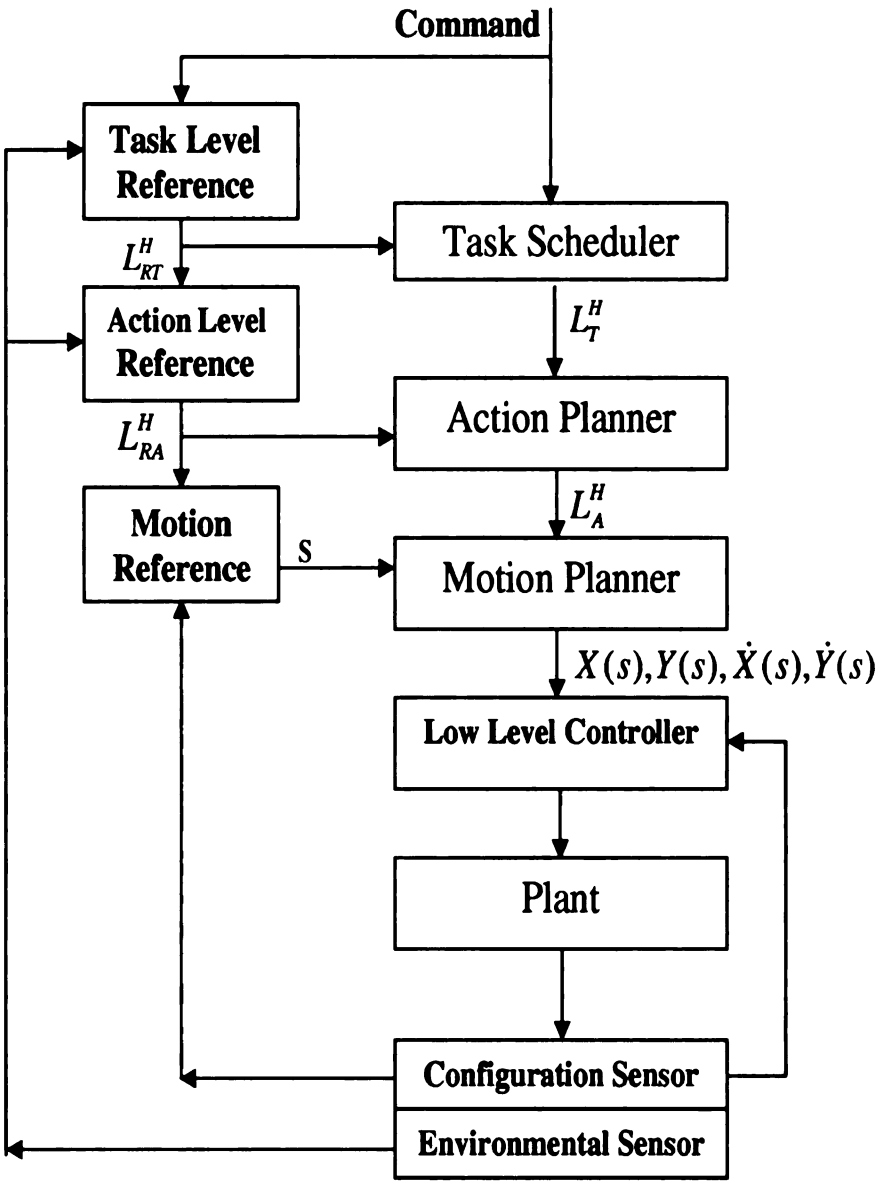


Figure 4.2: An Perceptive reference-based Framework for Robotic Systems.

4.2.1 Languages and Automata

Several concepts for the hybrid system model in perceptive frame are introduced to discuss hierarchical architecture of the system.

An Action Set consists of all the action primitives the robot can execute, for example, moving a robot, picking up/placing a part, open a gripper, and so on. The action set can be denoted as $A = \{A_1, A_2...A_{n1}\}$ where $A_i, i = 1, 2, ...n_1$ are action primitives.

A Task set is a set of all the task primitives for a robot system, each of which may include a few actions. The task set can be denoted as $T = \{T_1, T_2...T_{n2}\}$, where $T_i, i = 1, 2, ...n_2$ are task primitives. Usually, a task consists of several actions. For example, the task " pick up" consists of the actions that include "Go To" and "Close gripper".

An Action Reference Set is a subset of the action set. The set can be written in the form of $R_{Action} = \{r_{a1}, r_{a2}, ...r_{an3}\}$, where $r_{ai}, i = 1, 2, ...n_3$ are the references which govern the evolution of corresponding tasks.

The Task Reference Set is a subset of the task set and related to motions. It can be described as $R_{Task} = \{r_{t1}, r_{t2}, ...r_{tn4}\}$, where $r_{ti}, i = 1, 2, ...n_4$ are the references which govern the evolution of corresponding tasks.

The Motion Reference s is a continuous reference chosen for lower level path planning and control. For example, the distance the robot traveled, can be chosen as the motion reference. Examples of design and analysis of the mobile robot motion planning and control based on the continuous motion reference can be seen in [60].

The formal languages L_A, L_T, L_{RA} and L_{RT} can be built by the alphabets $\Sigma_A, \Sigma_T, \Sigma_{RA}$, and Σ_{RT} that are generated by the discrete variable sets $A, T, R_{Action}, R_{Task}$, respectively.

The relationships are shown as follows [28]:

$$\Sigma_{RA} = R_{Action} \tag{4.1}$$

$$\Sigma_{RT} = R_{Task} \tag{4.2}$$

$$\Sigma_A = A \tag{4.3}$$

$$\Sigma_T = T \tag{4.4}$$

$$L_A \subseteq \Sigma_A^* \tag{4.5}$$

$$L_T \subseteq \Sigma_T^* \tag{4.6}$$

$$L_{RA} \subseteq \Sigma_{RA}^* \tag{4.7}$$

$$L_{RT} \subseteq \Sigma_{RT}^* \tag{4.8}$$

Definition (Hybrid Language). A hybrid language is generated from an original alphabet Σ, consisting of discrete atoms, and numbers $\mathrm{N} \in R^n$. A word in the hybrid language can be described as an atom in the alphabet followed by numbers. The alphabet of the hybrid language is an extended alphabet of a formal language. A hybrid language is defined as the set of all the strings over the extended alphabet Σ^H. In other words, a hybrid language is the concatenation of the elements lying in the product set of the original alphabet and the Euclidean spaces. A hybrid language derived from the extended alphabet Σ^H is denoted L^H.

For example, if the alphabet of the hybrid language is the task set T, then the word in a hybrid language L_T^H can be "$T_1(x, y)$" meaning "going to the point(x,y)". The alphabet gives the qualitative description, the numeric part is the quantization of the linguistic expressions.

Definition (perceptive Automaton). Perceptive automaton can be defined as a tuple

$$M_e = (Q, \Sigma, \Sigma_R, \delta, Q_0, \eta, O). \tag{4.9}$$

where Q is the set of the discrete states of the automaton and Σ is the discrete control input set. Σ_R is the discrete perceptive reference set. Q_0 is a set of starting states and δ is the evolutions of states. η is the output function. O is output, it can be a vector.

Due to the perceptive specification, the following properties describe the state transitions and the output action trigged by the control command input and the reference inputs.

$$q_j = \delta(q_i, \sigma_j), \tag{4.10}$$

$$O_k = \eta(q_i, \sigma_k) \tag{4.11}$$

where $q_i, q_j \in Q, O_k \in O, \sigma_j \in \Sigma, \sigma_k \in \Sigma_R$.

The perceptive automaton improves the diversity of the input signals. Corresponding different inputs the automaton can have different responses including state switching and command issuing. The perceptive automaton is based on discrete perceptive reference. In order to combine the continuous perceptive reference and the discrete perceptive reference mentioned above, a hybrid automaton is capable of bridging the two kinds of variables. Based on [17][36], a hybrid automaton can be defined as follows.

Definition (Hybrid Automaton). A hybrid automaton is considered to be a tuple

$$M^H = (Q, X, \Sigma, \delta, Q_0, \eta, 0), \tag{4.12}$$

where Q is the set of discrete variables, X is the set of continuous variables, Σ is the alphabet of the discrete input, Q_0 is a set of starting states, η is the output function and O is the output. O can be a vector space.

The state transition function and the output function can be described as

$$q_j = \delta(q_i, \sigma_j, X), \delta_j \in \Sigma \tag{4.13}$$

$$O_k = \eta(q_i, \sigma_i, X) \tag{4.14}$$

Definition (Hierarchical Automaton). A hierarchical automaton is an automaton consisting of several nodes, some of which are other automata. The first automaton is an upper automaton, the other automata denoting the nodes of the upper automaton are called embedded automata. Then this architecture can be described as a tuple, $M_h = (Q_U, Q_{EM}, \Sigma_U, \Sigma_{EM}, \delta_U, \delta_{EM}, Q_{U0}, Q_{EM0}, O)$,

Σ_U is the input set for the upper automaton, Σ_{EM} is the input set of embedded automaton, Q_U is a set of the states(the nodes of the upper automaton), Q_{EM} is a set of the states of the automata embedded into nodes of Q_U, Q_{U0} is the set of the starting states of Q_U, Q_{EM0} is the set of the starting states of Q_{EM}. δ_U and δ_{EM} describe the transition functions of Q_U and Q_{EM}, respectively.

perceptive Automaton, hybrid Automaton, and hierarchical Automaton describe three basic properties of automata based on which the hybrid system framework are developed. The automata in the framework are thought of as the combinations of the above three basic prototypes.

4.2.2 Hierarchical Task Execution Architecture

As shown in Figure 4.2, task execution for a typical mobile robot system is implemented as a Schedule-Plan-Control Architecture. The hierarchical task execution architecture includes three parts: a task scheduler, an action planner, and a motion planner. These three parts are modeled using automata.

Task Scheduler

The task scheduler sequentially generates tasks according to the commands and discrete task level references. It is an automaton based on perceptive reference with hybrid variables.

$M_{TaskSch} = (Q, \Sigma_T^H, \Sigma_{RT}^H, X, \delta, Q_0, \eta, O_{TaskSch})$,

where the unexpected event input and task level reference input are in the language

L^H_{RT}, and L^H_T, respectively. The inputs from the Task Level Reference automaton trigger the task output in the hybrid language L^H_T. Logically, it chooses the higher priority task from the inputs, e.g. obstacle avoidance.

Action planner

The action planner is placed between the task scheduler and the motion planner in the architecture. The function of the planner is to generate a sequence of actions according to the task input from the task level. According to the definitions given above, the action planner is modeled as a hybrid perceptive automaton which can be described as a tuple,

$M_{ActPlan} = (Q, \Sigma^H_T, \Sigma^H_{RA}, X, \delta, Q_0, \eta, O_{ActPlan})$.

It has the perceptive reference inputs in the hybrid language L^H_{RA} and the task inputs in a hybrid language L^H_T denoted as:

$$I_{ActPlan1} = L^H_T \tag{4.15}$$

$$I_{ActPlan2} = L^H_{RA} \tag{4.16}$$

The output $O_{ActPlan}$ is in the language L^H_A.

The continuous variables in set X carried out by hybrid language inputs give the automaton continuous evolutions and can also be used to generate the outputs in the hybrid language.

The Action Planner has the transition function and the output functions as follows

$$q_{ActPlanj} = \delta(q_{ActPlani}, \sigma_j), \sigma_j \in \Sigma^H_T \tag{4.17}$$

$$O_{ActPlank} = \eta(q_{ActPlani}, \sigma_k), \sigma_k \in \Sigma^H_{RA} \tag{4.18}$$

The output can be a vector including several actions triggered by the perceptive

reference. The perceptive reference input triggers the output, and the task inputs cause the state transition of the automaton.

Motion planner

Generally, the motion planner is a mechanism to generate the continuous trajectory of the designed motion, based on the action sequence from the action planner. The motion planner in Fig. 4.2 is an automaton that is perceptive reference triggered and involves hybrid variables. We describe it as a tuple

$$M_{MotPlan} = (Q, \Sigma_A^H, s, X, \delta, Q_0, \eta, O_{MotPlan})$$

The continuous reference input s is the only variable of the continuous output function. Another input of the automaton is in the hybrid language L_A^H from the alphabet Σ_A^H. According to the discrete part of the language input σ in L_A^H, the automaton switches to the appropriate node. Each node of the automaton is a dynamic system (or a controller) which can issue the planned continuous trajectory for the lower level controller.

Therefore, the state transition functions and the output functions are:

$$q_{MotPlanj} = \delta(q_{MotPlani}, \sigma_j), \sigma_j \in \Sigma_A \tag{4.19}$$

$$O_{MotPlank} = \eta(q_{MotPlani}, s) \tag{4.20}$$

It can be seen that the state transition and output are triggered separately by two different inputs. For motion actions, the output to the robot controller can be a vector in the form of $O_{MotPlank} = [X(s), Y(s), \dot{X}(s), \dot{Y}(s)]^T$ for generating a complete motion trajectory.

4.2.3 Reference Generation

Three references are used in the framework, namely the continuous motion reference, the discrete action level reference, and the discrete task level reference.

The system has configuration sensors and environmental sensors. According to the architecture shown above, the environmental sensory measurements are used by the higher levels for discrete perceptive reference generation, and the motion reference can be issued based on the configuration sensor measurements.

In this subsection, the automata designed for reference generation are described.

Action level reference

The Action Level Reference automaton has a hierarchical embedded structure consisting of the upper automaton and the embedded automata that are nodes of the upper automaton. 4.4 This automaton receives the inputs from the task level reference automaton and the environmental sensors.

The upper automaton and the embedded automata respond to the task inputs and sensor inputs respectively.

The upper automaton has an architecture as shown in Figure 4.4. Each task input can change the states of the automaton which can be described as:

$$q_i = \delta(q_j, \sigma_i), j \neq i, \sigma_i \in \Sigma_T^H \tag{4.21}$$

In this case, each node of the upper automaton can be thought of as an automaton embedded in the node. The embedded automata, corresponding to different nodes in the upper automaton, may be different in state transitions and output generations. Figure 4.4 shows the architecture of an embedded automaton.

The states and the outputs can be changed based on environmental sensor measurements, i.e. Environmental Conditions(EnvCon), as shown in Figure 4.4.

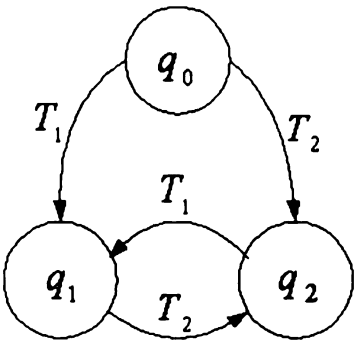


Figure 4.3: The Upper Automaton of the Action Level Reference automaton.

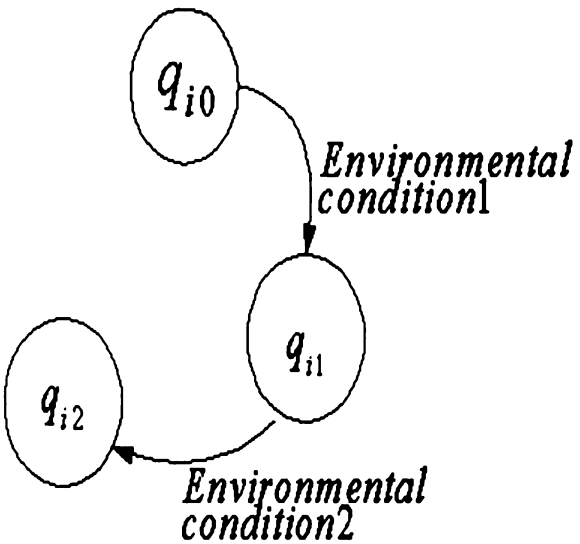


Figure 4.4: An Embedded Automaton of the Action Level Reference automaton.

The properties of such an embedded automaton can be described as

$$o_{ik} = \eta(q_{ij}, EnvCon_k = TRUE), j \neq k \tag{4.22}$$

$$q_{ik} = \delta(q_{ij}, EnvCon_k = TRUE), j \neq k \tag{4.23}$$

Where o_k is the output to the motion reference automaton and the task scheduler, which resets the motion reference and triggers the next action.

Task level reference

The Task Level Reference can be generated by automaton whose state transitions and output are triggered by the change of the environment. For example, Figure 4.5 shows how the automaton works while a mobile robot is approaching an unexpected obstacle. The automaton operates by issuing a proper command response to the unexpected event.

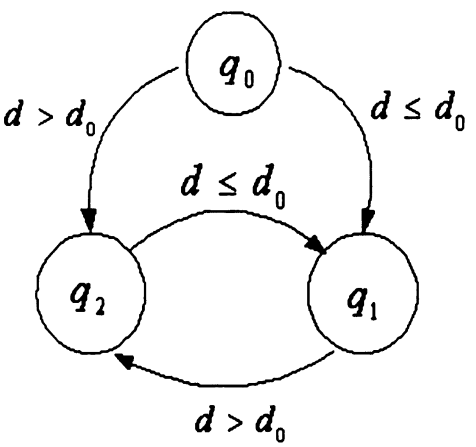


Figure 4.5: An Example of the Task Level Reference Automaton.

The transition function and the output function of the automaton are:

$$o_i = \eta(q_1 : \sigma_i) = AO(.), \sigma_i \in \Sigma_T^H \tag{4.24}$$

$$o_i = \eta(q_2 : \sigma_i) = \sigma_i, \sigma_i \in \Sigma_T^H \tag{4.25}$$

Where AO denotes Obstacle Avoidance, d is the distance between the mobile robot and its nearest obstacle; and d_0 is the minimum safe distance between them. The outputs in language L_{RT}^H go to Task Scheduler and the Action Level Reference automaton, respectively.

The robot controller design based on a continuous motion reference is referred to [60]. Essentially, the nonlinear feedback control law is given by [60] as

$$\tau = D(q)J_h^{-1}[\ddot{Y}^d(s) + K_v\dot{e}(s) + K_P e(s) - \dot{J}_h\dot{q}] + C(q,\dot{q}) + G(q) \tag{4.26}$$

The dynamics of the robotic system with continuous perceptive reference is described as

$$\frac{dw}{ds} = 2a \tag{4.27}$$

$$\frac{da}{ds} = u \tag{4.28}$$

4.3 Perceptive Control for Mobile Manipulation and Tele-Operation

A typical application of the proposed model is the execution of a scheduling-planning-control task in mobile manipulation. The task is to pick up an object and transport it to a desired destination with a mobile manipulator. In the route of transporting the object, an unexpected obstacle occurs. In order to complete the task, the mobile manipulator will execute an obstacle avoidance task. The mobile manipulator will then resume its original task and reach the destination.

4.3.1 Hybrid Languages

According to definition 6, the hybrid language can be composed of an extended alphabet. Each word contains the primitive discrete atom followed by the values of continuous variables. The task set can be defined as $T = \{T_1, T_2\}$. T_1 means "transporting", T_2 means "avoiding an obstacle." As defined above, the action set is $A = \{A_1, A_2\}$, where A_1 means "following a straight line," and $A2$ denotes "going through an arc."

To build the hybrid languages for the task set and action set, some continuous variables can be added to describe the atoms. In the extended task set, $T1(m1, n1)(m2, n2)$ denotes "transporting an object from point(m1,n1) to point(m2,n2)." $T2(m, n, l_O)$ denotes "to avoid an obstacle at(m, n) at the distance l_O." In the extended action set, the A1(m,n,l) denotes "following a straight line with directional cosine (m,n) and length l." The A2(m,n,r) denotes "go through an arc centered at (m, n) with length r." A3 and A4 denote "close the gripper" and "open the gripper", respectively.

4.3.2 Task Execution

Task Scheduler: The output of the task scheduler is $T1(m1, n1)(m2, n2)$ until the obstacle has been detected.

Action Planner: Based on the input $T1(m1, n1)(m2, n2)$, the transition and the output of the action planner are $q_{AP1} = (q_0, T1)$ and $o_{AP1} = \eta(q_{AP1}, reset) = A1$. When the planner has the action reference input A1 meaning that the A1 action is finished, the output action triggered by A1 is a vector $o_{AP1} = \eta(q_{AP1}, A1) = [A2, A3]^T$. The output triggered by action reference A11 is another action $o_{AP1} = \eta(q_{AP2}, A11) = A4$. The directional cosines and the length between A1 and A11 can be found with the parameters in T1 geometrically.

Motion Planner: The inputs $A1(m1, n1, l1)$, $A11(m11, n11, l11) \in \Sigma_A^H$ cause the states transition of the motion planner. $q_{MotPlan1} = \delta(q_0, A1)$, $q_{MotPlan11} = \delta(q_0, A11)$. The states denote different trajectories, ex., straight line path and arc path. Corresponding to the continuous perceptive reference input, s, the distance the robot traveled, the continuous trajectory outputs can be described by the following equations: for a straight line path

$$\dot{x} = mV(s) \tag{4.29}$$

$$\dot{y} = nV(s) \tag{4.30}$$

for a circular Path(an arc)

$$x = m + rcos(s/r) \tag{4.31}$$

$$y = n + rsin(s/r) \tag{4.32}$$

$$\dot{x} = -\frac{y}{r}V(s) \tag{4.33}$$

$$\dot{y} = \frac{x}{r}V(s) \tag{4.34}$$

In this case, the trajectory is made up of two pieces of straight lines $A1$ and $A11$.

Action Level Reference: The nodes of the upper automaton denote different paths, namely two segments of straight lines managed by the same procedure. The state transition is $q_{AR1} = \delta_{AR}(q_{AR0}, T1(m1, n1)(m2, n2))$. The output function and state transition of embedded automaton M_{AR1} can be described as $O_{AR11} = \eta(q_{AR10}, L \geq l_1)$ and $q_{AR11} = \delta_{AR1}(q_{AR10}, L \geq l_1)$, respectively. L is the length the robot traveled since the new action started.

4.4 Description for Language-Based Hybrid Perceptive Reference

In order to study the properties of the proposed model, the hierarchical architecture is considered as an integrated system. to describe the robotic manipulation system in the perceptive frame, Hybrid Perceptive Reference space is introduced in this dissertation.

4.4.1 Hybrid Perceptive Reference Space of Robotic Manipulation System

Considering the traditional time based systems, the reference has two significant features. First, it is a variable in a metric space. Second, it evolves itself forward

during the operation of the system.

The system illustrated in Figure 4.2, the hybrid system model, can be described with perceptive references and states, which can be expressed in a hybrid fashion. A hybrid perceptive trajectory is a finite or infinite sequence of reference interval $e = \{S_{TRi}S_{ARj}S_K\}$, where $S_K \in [0, s'_k] = I_{sk}, S_{TRi} \in \Sigma^H_{RT}, S_{ARj} \in \Sigma^H_{RA}$. Although the first two parts of $e = \{S_{TRi}S_{ARj}S_K\}$ have the continuous components, these components are stationary. The perceptive references at the task level and the action level can be still thought of as the discrete parts. The item s_K is a continuous variable.

4.4.2 Hybrid Metric Space

For the discrete topology, the hybrid metric can be generated in the following fashion:

$d(S_{TRi1}S_{ARj1}, S_{TRi2}S_{ARj2}) = 2$ if $S_{TRi1} \neq S_{TRi2}$,

$d(S_{TRi1}S_{ARj1}, S_{TRi2}S_{ARj2}) = 1$ if $S_{TRi1}S_{ARj1} = S_{TRi2}S_{ARj2}, S_{ARj1} \neq S_{ARj2}$

For the continuous item s_K, the metric is $d(s_{Ki}, s_{Kj}) = \| s_{Ki} - s_{Kj} \|$

Claim: the product space $S_{TR} \times S_{AR} \times S$ can be generated by the matric:

$d_D(S_{TRi1}S_{ARj1}S_{K1}, S_{TRi2}S_{ARj2}S_{K2}) = d(S_{TRi1}S_{ARj1}, S_{TRi2}S_{ARj2}) + \| s_{Ki} - s_{Kj} \|$.

Proof:

(1) "d" can satisfy the metric symmetric axiom ;

(2) $d_D(S_{TRi}S_{ARj}S_K, S_{TRi}S_{ARj}S_K) =$

$d(S_{TRi}S_{ARj}, S_{TRi}S_{ARj}) + \| s_K - s_K \| = 0$;

(3) It satisfies the triangle inequality.

For the discrete parts:

Let $d_D(1,2)$ denote $d_D(S_{TRi1}S_{ARj1}, S_{TRi2}S_{ARj2})$,

$d_D(2,3)$ denote $d_D(S_{TRi2}S_{ARj2}, S_{TRi3}S_{ARj3})$,

and $d_D(1,3)$ denote $d_D(S_{TRi1}S_{ARj1}, S_{TRi3}S_{ARj3})$.

If $d_D(1,2) = 2$, $d_D(2,3) = 2$. Then $d_D(1,3) \leq 2$, $d_D(1,3) \leq d_D(1,2) + d_D(2,3) = 4$;

If $d_D(1,2) = 1$, $d_D(2,3) = 2$. Then $d_D(1,3) = 2$, $d_D(1,3) \leq d_D(1,2) + d_D(2,3) = 3$;

If $d_D(1,2) = 1$, $d_D(2,3) = 1$. Then $d_D(1,3) \leq 1$, $d_D(1,3) \leq d_D(1,2) + d_D(2,3) = 2$;

If $d_D(1,2) = 0$, $d_D(2,3) = 1$. Then $d_D(1,3) = 1$, $d_D(1,3) \leq d_D(1,2) + d_D(2,3) = 1$;

If $d_D(1,2) = 0$, $d_D(2,3) = 0$. Then $d_D(1,3) = 0$, $d_D(1,3) \leq d_D(1,2) + d_D(2,3) = 0$;

For the continuous parts, the triangle inequality can be satisfied. Thus, d is a metric.

4.4.3 Independent Evolution

The perceptive reference is working as a "clock" or a "driver ", to trigger the planning and control activities. The triggering reference should evolve independently along a chain, an evolving chain, which plays the role of the time in the traditional time-based systems.

The "time" can be described as an ordered set. There exists a subset of the partially ordered set which is a totally ordered set.

We define the binary relation "$\preceq$", for perceptive reference value a and b, we say $a \preceq b$ if the time at which event a happens precedes to the time at which b happens. It can be proven that the perceptive reference trajectory is ordered by the binary relation "$\preceq$".

In an execution of a given task, the continuous reference s is a monotonic increasing function of time. Then the continuous part of the reference is ordered. Based on the sensor information and the task level linguistic input, the output of task level reference is ordered by the task sequence.

From the model of the action level reference $q_i = \delta(q_j, \sigma_i)$, it can be seen that the output of this level follows the order of the order of the state transitions on the upper automaton.

It means that if $S_{TRi1} \preceq S_{TRi2}$, then $S_{TR1}S_{ARj}S_{Kj} \preceq S_{TR2}S_{ARi}S_{Ki}$ for any i and j.

Given $a, b \in \{S_{TRi}S_{ARj}S_K\}$, the reflexivity and antisymmetry can be satisfied.

The transitivity can be stated by the ordering characteristics of the three level references. Finally, for any $a, b \in \{S_{TRi} S_{ARj} S_K\}$, satisfy the relation $a \preceq b$ or $b \preceq a$.

A perceptive reference trajectory is $\{S_{TRi} S_{ARj} S_K\}$, a totally ordered set with the binary relation $\preceq$.

Therefore, the perceptive reference can evolve independently over time.

4.5 Hybrid State Space Model for Robotic Manipulations

We denote the state of the integrated system in a hybrid way:

Let $X^h = \{q_T q_A q_M X\}$ be a hybrid state, where q_T is the state of task scheduler, q_A is the state of action planner, q_M is the state of motion planner. q_T,q_T, and q_T are discrete variables, X is the continuous state of the low level control system. Similar to the perceptive reference description, the metric of the state space can be defined by

$d_D(q_{T1} q_{A1} q_{M1}, q_{T2} q_{A2} q_{M1}) = 3$ if $q_{T1} \neq q_{T2}$,

$d_D(q_{T1} q_{A1} q_{M1}, q_{T2} q_{A2} q_{M1}) = 2$ if $q_{T1} = q_{T2}, q_{A1} \neq q_{A2}$

$d_D(q_{T1} q_{A1} q_{M1}, q_{T2} q_{A2} q_{M1}) = 1$ if $q_{T1} q_{A1} = q_{T2} q_{A2}, q_{M1} \neq q_{M2}$

$d(q_{H1}, q_{H2}) = d_D(q_{T1} q_{A1} q_{M1}, q_{T2} q_{A2} q_{M1}) + \| X_1 - X_2 \|$

We denote the hybrid system as $dX^h/ds^h = f(X^h, u(s^h))$. Therefore, the entire system can be described in the hybrid space.

CHAPTER 5

ANALYSIS OF CONTROL SYSTEM IN PERCEPTIVE FRAME

5.1 Introduction

The proposed control method is described in the perceptive frame. The nature of the hybrid perceptive reference ensures the control model has the specifications from both the hybrid systems and the evolution of the perceptive references. Hybrid systems have been extensively investigated in the past. One of the limitations in the literature is that most of the discussions are time-based. On the other hand, it is desirable to develop the analysis methods for the dynamical properties of the hybrid switched system with hybrid perceptive reference and states.

Based on the description of the references and hybrid states of the integrated system. In this chapter, the analysis issues will be focusing on the evolution property of the perceptive references, stability in hybrid space, and instruction complexity during the unexpected event processing.

5.2 Evolving of Perceptive Reference

Any non-time reference is generated by the observation of the environment or the system states, which are independent from time. For instance, the distance the mobile robot traveled can be chosen as the motion reference. The evolution of the reference can stopped by the obstacle that blocks the motion of the system until the block is released. Thus, evolving is crucial for perceptive reference.

The system illustrated in Figure 4.2, the hybrid system model, can be described with perceptive references and states, which can be expressed in a hybrid fashion. A hybrid perceptive reference trajectory is a finite or infinite sequence of reference

interval $e = \{S_{TRi}S_{ARj}S_{ij}\}_{i,j}$, where $S_{ij} = [0, s_{ij}], S_{TRi} \in \Sigma_{RT}^{H}, S_{ARj} \in \Sigma_{RA}^{H}$. We denote s^h as the value of the hybrid reference, i.e. $s^h \in e$. Furthermore, an execution of a hybrid perceptive automaton describes a collection $\chi = \{e, q^M, x\}$, where e is a hybrid perceptive reference trajectory, q^M is a map from $S_{TRi}S_{ARj}$ to Q, and x maps the continuous reference s to the continuous state space. We define L_s is the minimum value of the sum of the distance, i.e., $L_s = \Sigma S_{ij}$. An execution is minimum if $L_s = min\{\Sigma S_{ij}\}$.

A hybrid system is called deterministic if the system has a unique minimum execution.

To simplify the problem, we impose the following standing assumption.

Assumption: For given initial states (q_0, x_0) and inputs, the minimum execution in the perceptive frame is unique for a given hybrid system, i.e., the execution is deterministic. Furthermore, the execution continuously depends on the initial state and inputs.

The system in the perceptive frame is driven by the evolution of the perceptive references. The hybrid perceptive references are introduced to keep the system evolving.

A discrete transition of the hybrid perceptive references occurs when an unexpected event blocks the evolution of the references. Such an event can occur at any state (q_b, x_b) corresponding to the perceptive reference $e_b = \{S_{TRb}S_{ARb}S_b\}$. It could lead the system to another state from which the execution has unique solution to the final state and the evolution of the perceptive references cannot be blocked by the unexpected event again.

This fact is stated more formally in theorem (Evolving) in the section of proposed work.

One of the major advantages of the use of linguistic control in perceptive frame is that the perceptive control method can modify the preplanned trajectory in the

presence of unexpected events. Generally, in the time-referenced systems, replanning the path is required as the system is blocked by unexpected events, it causes extra computation complexity in this respect. The control systems in perceptive frame are driven by perceptive references, in other words, for robotic systems, the path and control command generated according to the perceptive reference, once the path planning is accomplished, the desired path is triggered by the events in the perceptive references($X = P(s^h)$. It gives the system a large flexibility in planning and control. When the unexpected events block the evolution of the perceptive reference, the system can switch to avoid the blocks and eventually come back to the predefined perceptive reference, then resume the preplanned path without replanning.

Definition (Reference complexity): given a perceptive control system, and a linguistic description, the reference complexity of a given task T is given by the minimum length word $S_{TR1}S_{AR1}, S_{TR1}S_{AR2}...S_{TRi}S_{ARj}...$, it is denoted as $C(T)$. Moreover, the reference complexity can represent how many discrete switches the system should make when the system start from an initial state to a destination state.

This can be described based on the architecture shown in Figure 4.2. The state transition function and the output function for the action planner are:

$$q_{ActPlanj} = \delta(q_{ActPlani}, \sigma_j), \sigma_j \in \Sigma_T^H \tag{5.1}$$

$$O_{ActPlank} = \eta(q_{ActPlani}, \sigma_k), \sigma_k \in \Sigma_{RA}^H \tag{5.2}$$

The state transition functions for the motion planner

$$q_{MotPlanj} = \delta(q_{MotPlani}, \sigma_j), \sigma_j \in \Sigma_A \tag{5.3}$$

Intuitively, if the states are different in the task level, the system needs to switch and plan a new task for example, if the states will switch at the action level, it does not cause the transition on task level. then complexity could be reduced without the

task level scheduling.

Assume that for a given task, the average of the minimum length of the word is N. There exist N unknown static obstacles on the preplanned trajectory. The unexpected events make the mobile robot switch the discrete state to avoid the current segment of the preplanned execution. The instruction complexity is $N + M * b$, if the system replans the path every time, the total instruction complexity is $N + (N-1) + (N-2)...+$, the order of the complexity is N^2.

The proposed method is capable of dealing with unexpected events, which can block the perceptive reference. It is essential to describe and prove this proposition theoretically.

Proposition (Evolving): *Given any state* $(q_b, x_b) \in \chi_0 = \{e, q, x\}$, *on an execution of the hybrid automaton, there exists a discrete transition to* $q_t \in Q$, *the system has a unique minimum execution* $\chi_m = \{e, q, x\}$ *to the final state, and* $(q_b, x_b) \notin \chi_m = \{e, q, x\}$.

Roughly speaking, the proposition explains that since both the initial state and the final state are placed in the defined reachable domain, the system has a unique solution. The continuous dependence on initial conditions guarantees that the hybrid system cannot reach the state blocking the evolution of the continuous perceptive reference.

To investigate the ability of the proposed perceptive control model to reduce the complexity during the unexpected events is one of the future research issues. As the environment becomes increasingly complex, in particular, the dynamic obstacles may occur on the trajectory of the robot. The dynamic obstacles may block the mobile robot at any point with any possible velocities. It causes the extra instruction complexity in the perceptive frame. The perceptive frame control takes advantage of the sensor information which contains the knowledge of environmental changes.

In order to reduce the complexity, two approaches are considered. First, the

hierarchical architecture shows that if the perceptive reference of the system can conduct discrete transition on the lower level, the complexity can be reduced, because the transition on task level usually consists of several low level actions. Second, the complexity is largely depending on the event-triggering architecture. The perceptive reference has rich information about the current state of the system and environmental changes. How to employ the perceptive information in a complicated environment with multiple obstacles or dynamical obstacles, therefore, is crucial.

5.3 Stability of Perceptive Control System

As a dynamical system, the robotic manipulation system has to be stable. For mobile manipulation systems, the controller can run in different types, motion control of the mobile base or motion control of the robot arm, and autonomous control or teleoperation control. The segments in different types of control form the whole control procedure.

Lyapunov stability plays an important role in the control system analysis and design. In [63][60], for continuous perceptive systems, it has been proven that if a system is asymptotically stable with time t as its motion reference base, and if the new motion reference s is a (monotone increasing) non-decreasing function of time t, then the system is (asymptotically) stable with respect to the new motion reference base s. The time-based Lyapunov function $V(t)$, then can be described as $V(S)$, i.e., the Lyapunov function for motion reference based system.

For the hybrid model, an execution is a sequence of segments described by the perceptive reference trajectory and the corresponding hybrid states.

For the system $dx/ds = f(x)$, we say that $V(s)$ is a candidate Lyapunov function if $V(s)$ is a continuous positive definite function (about the origin 0) with continuous partial derivatives. Note this assumes $V(0) = 0$.

Considering switched hybrid systems, the systems operate in a hybrid metric

space. The stability property of such systems could be described in the hybrid state space of the perceptive frame. Therefore, multiple Lyapunov function can be used to discuss the stability issue.

Given a dynamical system in the perceptive frame, if there exist Lyapunov functions $V_1(s^h), V_2(s^h)...$ with hybrid perceptive references, corresponding to the hybrid perceptive reference trajectory and different segments of executions, for a given strictly increasing sequence of $S_{TRi}S_{ARj}$ (discrete reference values), denoted by $I = S_{TRi}S_{ARj}$ in the perceptive reference, we say that V is a *Lyapunov-like* function for function f and execution $\chi = \{e, q, x\}$, if

1. $dV(x(s^h))/ds \leq 0$
2. V is a monotonically non-increasing on ordered set I.

Theorem (Lyapunov Stability): *Suppose we have candidate Lyapunov functions V_i,$i = 1...N$, and vector field $dx/ds = f_i(x)$ with $f_i(0) = 0$, Let E be the set of all switching sequences associated with the system. If for each $s^h \in E$ we have for all i, V_i is Lyapunov-like for f_i. Then the system is stable in the sense of Lyapunov.*

Proof: Given a strictly increasing sequence of $S_{TRi}S_{ARj}$, we denote the time at which the reference happen as $\tau(S_{TRb}S_{ARb})$. Because the perceptive reference is an ordered set by relation $\preceq$, i.e., $\tau(s_a^h) \leq \tau(s_b^h)$, for any $s_a^h \preceq s_b^h$. Based on the stability theorem in [60], the corresponding time based functions have the identical monotony. From [6], for the case if $N = 2$, or $N > 2$, we always can pick a small neighborhood of the origin with ρ from that the trajectory will stay in $B(r)$, which is the minimum ball around the origin over all the possible Lyapunov functions.

When the system has only one vector field $dX/ds^h = f(X)$, for any given segment of execution $\chi = \{e, q, x\}$, the continuous reference evolves in the interval$[0, s]$ due to the discrete transitions of the discrete perceptive references. The system is still stable since Lyapunov-like functions keep the monotony properties under these circumstances. The strictly increasing sequence of $S_{TRi}S_{ARj}$ still exists. Hence the

Lyapunov stability can not be changed by switch of the references. The multiple Lyapunov-like functions provide a convenient way for analyzing the state stability of the switched systems and the further discussion on input-state stability.

5.3.1 Input-State Stability of Hybrid Perceptive Control

The given hybrid control model can be thought of as a forced control system. Hence, the input-state stability is required, in particular, for the hybrid switched systems. One of the future research issues is to study the input-state stability on the hybrid control model. Switched control inputs on the low level controller of the given system model can be thought of as piecewise continuous and bounded function of s_h.

Definition *The system above is said to be locally input-to-state stable if there exist a class KL function β, k_1 and k_2 such that for any initial state $x(s_0^h)$ with $\| x(s_0^h \| \leq k_1$ and any input $u(s^h)$ with $sup_{s^h \geq s_0^h} \| u(s^h) \| < k_2$, i.e., for any bounded signal. The execution on the system exists and satisfies*

$$\| X(s^h) \| \leq \beta(\| x(s_0^h) \|, d_D(h^s, s_0^h)) + \gamma(sup \| u(s) \|) \tag{5.4}$$

It is said to be input-to-state stable if the above inequality is satisfied for any initial state and any bounded input $u(s^h)$

Based on the definition, we will discuss the input-to-state stability of the system with the hybrid perceptive reference, when the corresponding time based system have input-to-state stable for the given conditions on the time-based input.

The input-to-state stability can give an approach to analyze the system stability. In particular, the hybrid systems have to deal with the control signal with different switched segments due to and different control types including teleoperation and autonomous control. different control inputs require the input-to-state stability. Control

signal can be tracked. input-to-state stability can guarantee that the control signal can be tracked by the hybrid system in the perceptive frame.

Theorem 1 (Stability): *Suppose we have candidate Lyapunov functions* V_i,$i = 1...N$, *and vector field* $dx/ds = f_i(x)$ *with* $f_i(0) = 0$, *Let* E *be the set of all switching sequences associated with the system. If for each* $s^h \in E$ *we have for all* i, V_i *is Lyapunov-like for* f_i. *Then the system is stable in the sense of Lyapunov.*

Proof: Given a strictly increasing sequence of $S_{TRi}S_{ARj}$, we denote the time at which the reference happen as $\tau(S_{TRb}S_{ARb})$. Since the perceptive reference is an ordered set by relation $\preceq$, i.e., $\tau(s_a^h) \leq \tau(s_b^h)$, for any $s_a^h \preceq s_b^h$. Based on the stability theorem for continuous perceptive system, the corresponding time based functions have the identical monotony. According to [5] for the case if $N = 2$, or $N > 2$, we always can pick a small neighborhood of the origin with ρ from that the trajectory will stay in $B(r)$, which is the minimum ball around the origin over all the possible Lyapunov functions.

Theorem 2(Exponential Stability): *Suppose we have candidate Lyapunov functions* V_i,$i = 1...N$, *and vector field* $dx/ds = f_i(x)$ *with* $f_i(0) = 0$, *Let* E *be the set of all switching sequences associated with the system. In the perceptive frame, if for each* $s^h \in E$ *we have for all* i, V_i *is Lyapunov-like for* f_i. *Then the system is exponentialy stable in the sense of Lyapunov. If There exist constants* $\alpha_g, \beta_g, \gamma_g, g = 1, ..., l$, *such that:*

$$-- \quad \forall x \in \Omega, \alpha_g(\| x \|) \leq V_g(x) \leq \beta_g(\| x \|) \tag{5.5}$$

$$-- \quad \forall x \in \Omega,$$

$$-- \quad \forall x \in E_{gr}, V_r(x) \leq V_g(x). \tag{5.6}$$

for switching from q to r.

Proof: Since the perceptive reference can be introduced in to the hybrid system.

The Lyapunov function V_r has the reference segment, which evolves from 0, after discrete transitions. Then in the given case, $V_r(x) \leq V_r(x(t_1))/(-\gamma_r/\beta_r(s))$, Since $\forall x \in E_{gr}, V_r(x) \leq V_g(x)$ for switching from g to r.

$$\| x(t) \| \leq \delta e^{-N} \| x(s^h) \| . \tag{5.7}$$

$$N = \delta(s - s^h) \tag{5.8}$$

Where perceptive reference s^h is a discrete transition. Therefore the system is exponentially stable in sense of Lyapunov.

5.3.2 Continuity of input-output mapping

The offset on the continuous variables of the task level linguistic inputs. For the tasks, which have the same discrete task but different continuous variables, i.e., two task controls in the same task subspace, if the continuous parts approximate sufficiently, then the states of the system are sufficiently close to each other.

It can be formally described by followings:

For a given task subspace T_i, the mapping L is referred to as a continuous mapping, if there always exists an ϵ, for given δ such that $\| \Delta A \| < \epsilon$, when $\| \Delta T \| < \delta$. The intuitive meaning is that both the task and the action will evolve continuously. The norm $\| \Delta A \|$ is referred to as the sum of all the action offset. I.e., $\| \Delta A \| = \Sigma \| \Delta A_i \|$. Thus, the input and output of the task controller in the linear space T and A are continuous dependent.

5.3.3 Stability conditions for switched systems

The stability conditions include two parts. First, the switching between the task subspace with different dimension is stable under certain conditions of Lyapunov

functions. Second, the continuous variable of the task input do not spoil the stability, if for a given value, the system is stable.

Assumption: The input and the output of the task controllers in linear space are continuous dependent.

Claim 3: If the vector fields corresponding to the switching task T1 and T2 in Ω_1 and Ω_2 are exponentially stable. Then given any continous part T1 and T2, the switching tasks will be stable.

Proof: The control input of the low level controller is generated by linear mappings $L : T- > A$ and $L' : A- > \Omega$, so the mapping $L(L')$ follows lipschitz condition, such that $\| L(L')(x) - L(L')(y) \leq l \| x - y \|$.

For the conditions of the theorem 2 still hold, for a close loop controller, we have:

$$\forall \mathbf{x} \in \mathbf{\Omega_i}, \partial \mathbf{V_i}(\mathbf{x})/\partial \mathbf{s} = \partial \mathbf{V_i}(\mathbf{x})/\partial \mathbf{x} \cdot \mathbf{f}(\mathbf{x}, \mathbf{s}), \tag{5.9}$$

Then, from different vectors T in the same task subspace, Δu is defined as the pertubation the controller caused by the offset of the vector T1 and T2. Substitute linear mappings $L : T- > A$ and $L' : A- > \Omega$ and lipschitz coefficiant l.

$$\| x(t) \| \leq 1/l\delta \cdot (e^M) \| x(s_i^h) \| . \tag{5.10}$$

$$M = -\delta(s - s_i^h)/l \tag{5.11}$$

The system is stable in sense of Lyapunov during the task switches.

5.4 Modeling of Unexpected Event

The ability to deal with unexpected events is crucial for a perceptive control system. Q represents the set of the vector fields, $(q_1, ..., q_n)$, for a m-dimensional space, at

least it has m m-dimensional orthogonal vector fileds. $f_1 ... f_m$. Therefore, for any 2 points $x1$ and $x2$, the system can start from $x1$, $x2$ within finite time of switches over the vector fields.

In task execution, the continuous reference s should satisfies: $L_f S(q,x) > 0$, it guarantees that the continuous reference is an increasing function of the time t.

5.4.1 Unexpected Events(UE) in Robotic Manipulations

Unexpected events in robotic manipulations can be described as follows:

For a given $e = \{S_{TRi} S_{ARj} S_K\}$, where $S_K \in [S_k, s'_k], S_{TRi} \in \Sigma^H_{RT}, S_{ARj} \in \Sigma^H_{RA}$. and an execution of a hybrid perceptive automaton $\chi = \{e, q, x\}$, A unexpected event happends, when the following two condition hold:

$$ds/dt \mid_S = 0, S = s^h_u \in (s^h_0, s^h_f), \tag{5.12}$$

$$s^h_u \neq S_{TRI}, S_{ARJ}. \tag{5.13}$$

Furthermore, an unexpected event represents a convex region U_A that is an open set of states, in which the above conditions hold, the boundary is a scalar function $\omega(x) = 0$, and $\omega(x) > 0$ for $x \in U_A$, when UE happen, using "Lie Derivative", we have $L_f \omega(x) > 0, \omega(x) = 0$.

It is a local blocking event of Task T if we can find a subset of the trajectory $\chi_1 = \{s_t, q, x\}$, where $s_t = (s^h_u, s^h_f) \notin U_A$.

5.4.2 Unexpected Event Processing in Perceptive Frame

For hybrid automata, the unexpected events can be described as a disturbance for a system. The system is desired to be able to return to the designed trajectory after the disturbance. Or, the unexpected events can be treated as a task executed before

finishing the current task. The new task is to switch out of the blocking state then return to and resume the previous task. For the given model, the unexpected event will affect the states, causes a discrete state transition. A new vector field will be applied on the continuous states of the system. which is an orthogonal vector field to the old one, $< f_i(x) \cdot f_{i+1}(x) >= 0$, then it can satisfy the following conditions: $L_{f_{i+1}}\omega(x) < 0$, $L_{f_{i+1}}S(q,x) > 0$, the former makes the system leave domain U_A from the boundary $\omega(x) = 0$, the latter guarantees the evolution of the continuous reference.

The following theorem shows the existence of the solution at unexpected events in the perceptive frame.

Theorem 4: *if the unexpected event e_u is a local blocking event, there exists another finite automaton and with the same alphabet, corresponding to the new automaton, the automaton can go back to the original task trajectory $\chi = \{e, q, x\}$.*

The following is the sketch of the proof of the Theorem 4. Since the given UE is a local blocking event, $B - U_A$ is a path connected domain. There exists a sequence of vectors connecting x_{eu} and $x_s, s = s_{return} \in s_t$, which is $l_1 ... l_p$, p is a finite number. Since the vectors can be described as the trajectory in the given orthogonal vector fields though a linear mapping M, then the trajectory can be decomposed as a finite number of the connected segments in the switching vector fields. We denote the segments with the action level alphabet, as $w_1 ... w_m$, m is a finite number.

Based on the sequence of the segments, a new automaton can be built for UE processing and switching back to the original task. The string of the corresponding automaton is $w_1 ... w_m$, which is embedded in the string of original task execution.

5.4.3 Unexpected Event Processing: An Example

In Mobile Manipulation, after an unexpected obstacle is detected, the task reference generates the event of obstacle $T2(m_o, n_o, l_o)$, the output of the task scheduler is

$T2(m_o, n_o, l_o)$. It causes a discrete switch over discrete variables, an obstacle avoidance on the task execution to switch and resume the evolution of the continuous perceptive reference. When the effect of the unexpected event is removed, the robot will resume the previous task to reach the destination. The perceptive reference will never be blocked.

Action Planner: The transition function and the output function are $q_{AP2} = \delta(q_{AP1}, T2(m_o, n_o))$ and $o_{AP2} = \eta(q_{AP2}, reset) = A2(m_{a2}, n_{a2}, r_{a2})$, respectively.

Motion Planner: The motion planner receives the action $A2(m_{a2}, n_{a2}, r_{a2})$ from the action planner, the outputs can be described by equations(3.30) (3.33).

Action Level Reference: This automaton has a state transition on the upper automaton due to the unexpected event and returns to the previous state after the process of the unexpected event. The triggered procedure is the same as in the task execution process.

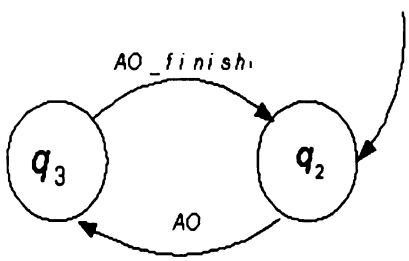


Figure 5.1: Unexpected Event Processing in the Task Level Reference.

The transition function and the output function are $q_{AP2} = \delta(q_{AP1}, T2(m_o, n_o))$ and $o_{AP2} = \eta(q_{AP2}, reset) = A2(m_{a2}, n_{a2}, r_{a2})$, respectively.

The motion planner receives the action $A2(m_{a2}, n_{a2}, r_{a2})$ from the action planner, the outputs can be described by equations(3.22)-(3.25).

This automaton has a state transition on the upper automaton due to the unexpected event and returns to the previous state after processing the unexpected event. The triggered procedure is the same as in the task execution process.

In real time tele-operated systems, it is important to synchronize the command

from the operator to the arm controller. The time delay of the communication between the operator and controller will significantly affect the control performance. If the controller cannot receive the next position or velocity command before it finishes executing the current one. It has to stop or just keep going on the current velocity, the former reduces the smoothness of execution while the later causes large control errors. In the hybrid perceptive frame model, the delayed command is thought of as an unexpected event on action level, that can prevent the discrete perceptive reference from evolving.

The routing for the unexpected event is to switch to another action which decelerates the motion to make the trajectory smooth until the next control command is received then switch back to the previous state and the path plan is modified. As the result, the hybrid perceptive reference cannot be blocked.

CHAPTER 6

EXPERIMENTAL RESULTS

6.1 Experimental System Setup

The experimental system, as shown in Figure 6.1, is an integrated control system. The hardware structure and the software structure of the system will be introduced in this chapter.

6.1.1 Hardware Structure

The robot controller is a computer control system. The mobile manipulator has two PCs for the Puma 560 control and the mobile robot. Both PCs in the mobile manipulator are communicating with the workstation or PC whose interface can have humans involved into the control process. Human intervention is involved through joystick controller. Two kinds of force-feedback joysticks are employed in the experimental system: the MicroSoft(MS) joystick, which can send the human motion command in velocities, and the Phantom joystick, which issues the motion command in positions.

The mobile manipulator has a laser range finder, which can measure the distance of the object ahead with a uniformly distributed bundle of laser lights. The encoders of the mobile manipulator can measure the position of gripper.

6.1.2 Software Structure

The control software of the experimental system consists of:

- **Robot Arm Control program**: A program for controlling the Puma 560.
- **Mobile Base Control program:** This is designed for motion control of the base and collection of sensor measurements.

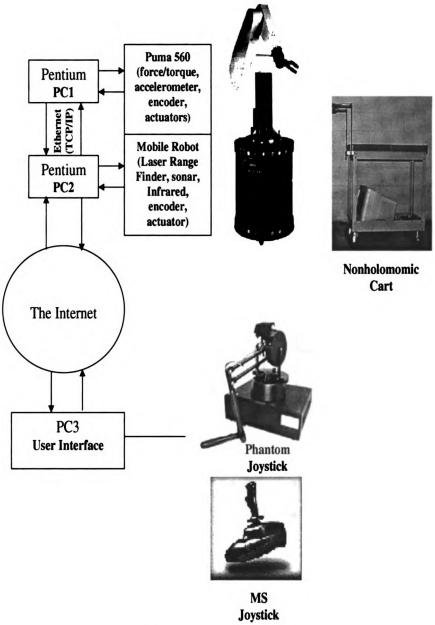


Figure 6.1: The Experimental System

- **Mobile Base Control Proxy:** It is a program to receive the command from the shared memory and send the commands to Mobile Base Controller in PC2.

- **Online Model Learning software:** Based on the measurements from the laser range finder, the encoders and the force/torque sensor, the parameters of the model for the cart can be estimated.

- **Hybrid Linguistic Control program:** This is the high level controller of the experimental system, running in the perceptive frame.

- **Joystick Client User Interface:** The interface software converts the measurements of human operation on the joystick into motion control signals and sends the control signals to a remote motion controller.

- **Joystick Control Server:** This program is running in PC1 and designed to receive the commands from Joystick Client User Interface, then send the commands to the shared memory in PC1

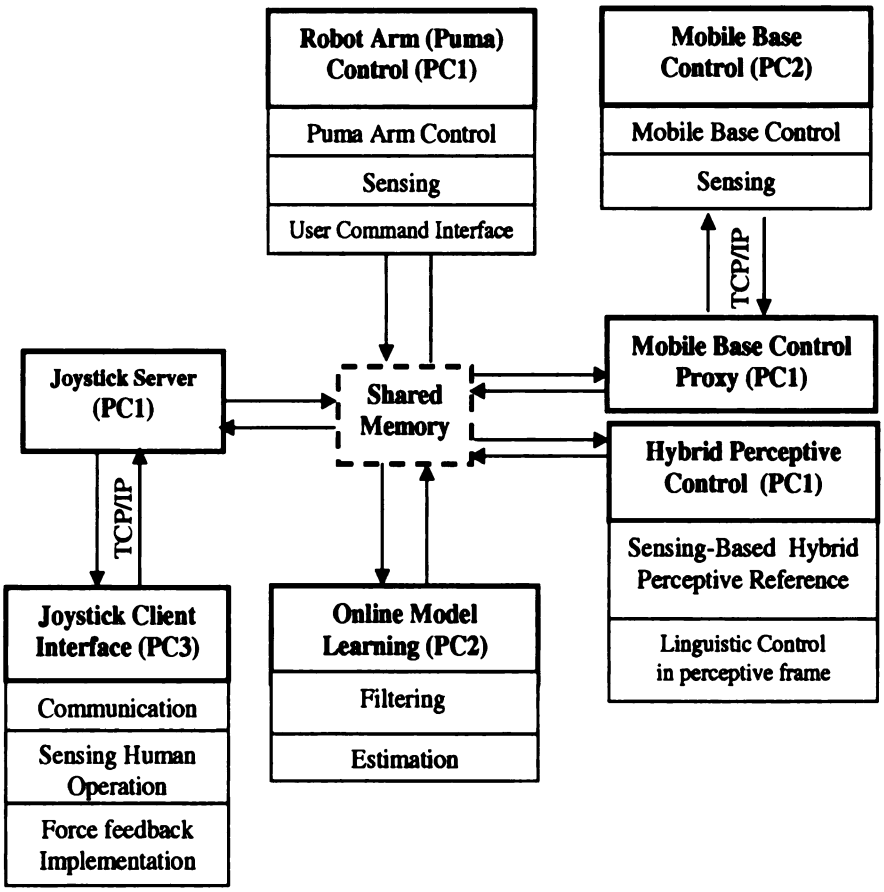


Figure 6.2: The Software Structure of the Experimental System

Figure 6.2 shows the modules in each program and the relations between the control programs, based on the hardware structure shown in Fig. 6.1.

6.1.3 *Robot Controller and Task Modeling*

The mobile manipulator can be controlled remotely by an operator through the joysticks or can run autonomously by a computer program.

In perceptive frame, the robot controller can be described as

$$\tau = D(q)J_h^{-1}[\ddot{Y}^d(s) + K_v\dot{e}(s) + K_P e(s) - \dot{J}_h\dot{q}] + C(q,\dot{q}) + G(q) \tag{6.1}$$

The dynamics of the robot is described as

$$\frac{dw}{ds} = 2a \tag{6.2}$$

$$\frac{da}{ds} = u \tag{6.3}$$

Where $W = v^2$. As shown in Figure 6.1, the task model is a nonholonomic wheeled cart. The cart has two omnidirectional wheels installed on the front of the cart, and two rear wheels which are mono-directional. A flat board is installed on the back of the cart. The laser range finder can detect the distances to the points on the board based on which the orientation can be estimated.

6.2 Experimental Results for Online Model Learning

The proposed model learning method has been tested using a Mobile Manipulation System consisting of a Nomadic XR4000 mobile robot, a puma560 robot arm attached on the mobile robot. A nonholonomic cart is gripped by the end-effector of the robot arm as shown in Fig.1. There are two PCs in the mobile platform, one uses Linux as the operating system for the mobile robot control and the other uses a real time operating system QNX for the control of the Puma560. The end-effector is equipped with a $Jr3$ force/torque sensor.

In order to identify the model of the cart, two types of interaction between mobile manipulator and the cart are planned. First, the robot pushes the cart back and forward without turning the cart. The sensory measurement of the acceleration and the force applied to the cart can be recorded. Second, the cart was turned left and right alternatively to obtain the sensory measurements of the position of the point A and the orientation of the cart.

To estimate the mass of the cart, Recursive LSM is used. The measured acceleration signal and the measured signal of the pushing force have independent white noise. Hence, the estimation should be unbiased. The estimate of the mass of the cart can be obtained directly by LSM.

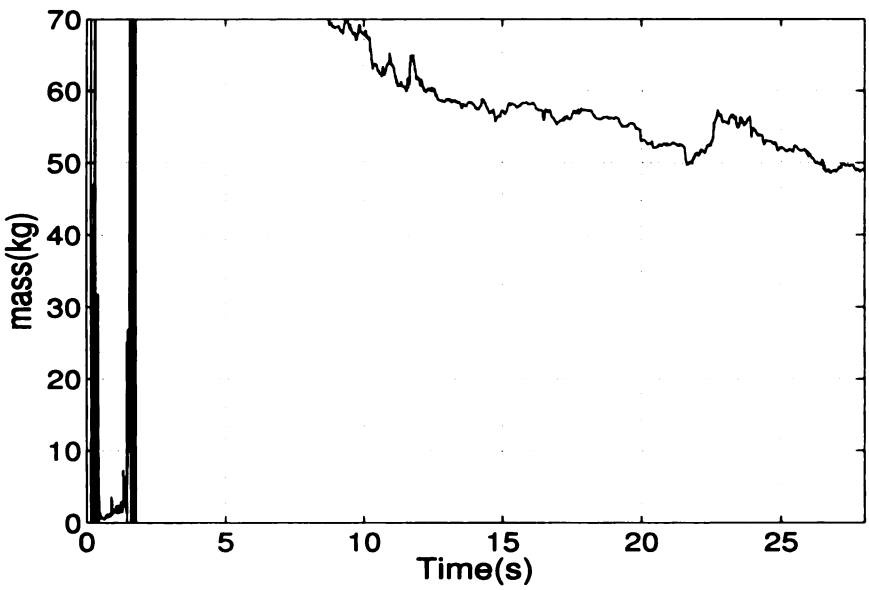


Figure 6.3: The mass estimation, for m=45 Kg.

Fig.6.3 and Fig.6.4 indicate the mass estimation process. In the beginning, the estimation is oscillating, however, a few seconds later, the estimation became stable. The mass estimation results are listed in Table 6.1.

According to the proposed method, the algorithm filters the raw signal to have different bandwidths. The experimental results are shown by the graphs below.

The figures(Fig.6.14, 6.15 and 6.16)indicate the $\tilde{e}$ and the parameter estimation

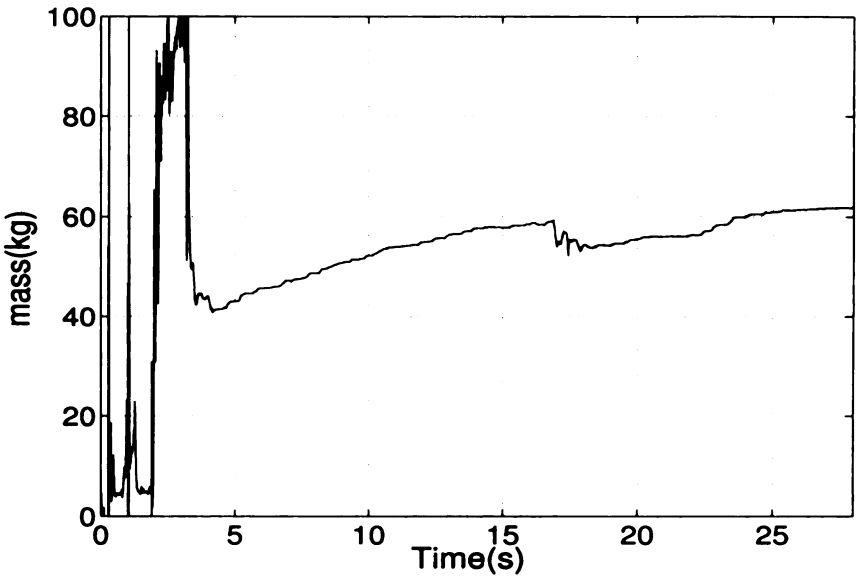


Figure 6.4: The mass estimation, for m=55 Kg.

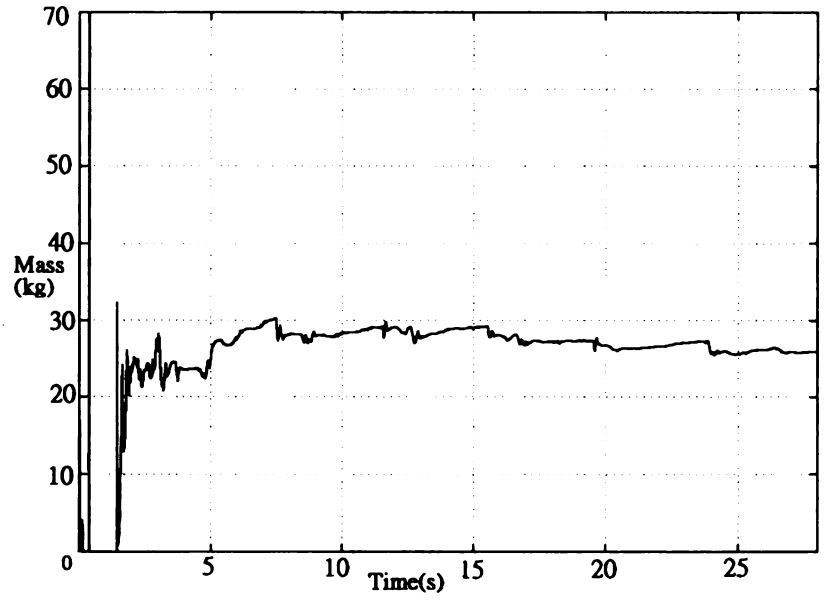


Figure 6.5: The mass estimation, for m=30 Kg.

errors in different levels, in case of L=0.93m, 1.31m, and 1.46m, respectively.

The horizontal axes represent the index of the estimation level, the level here means the estimation stage as shown in figure 6.14, 6.15 and 6.16. There is maximally 13 estimation stages in this estimation, therefore the index of the levels ranges from 1 to 13. The vertical axes of the figures represent the absolute value of relative

mass	*estimate*	*error*(*kg*)	*error*(%)
45.0	49.1	4.1	9.1%
55.0	62.2	7.2	13.1%
30.0	26.8	3.2	10.7%

Table 6.1: The Results of Mass Estimation.

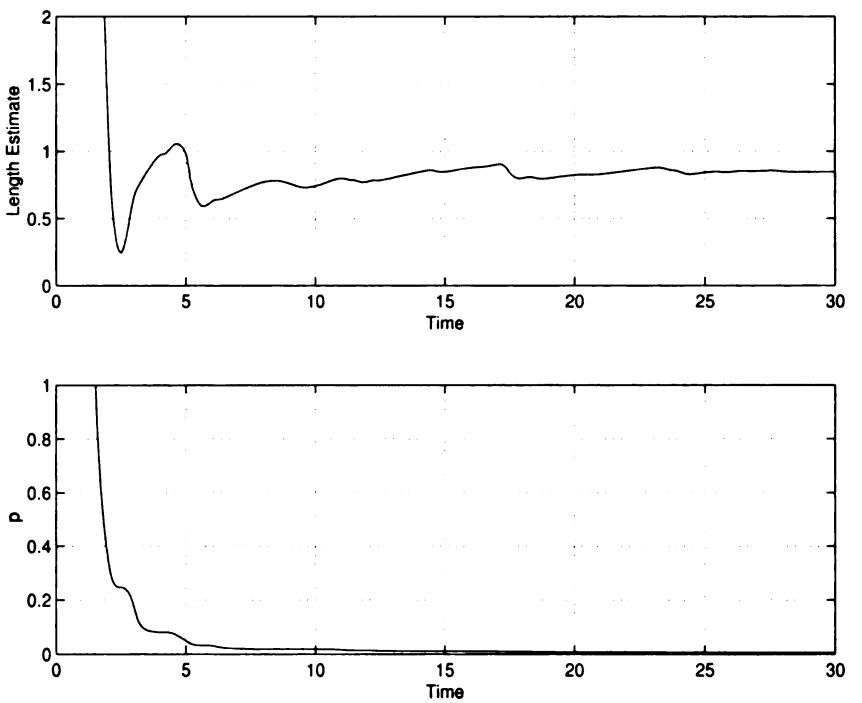


Figure 6.6: The length estimate and variance P at 9th levels for L=1.31m.

estimation error, and the value of $\tilde{e}$.

Fig. 6.6, Fig. 6.7, Fig. 6.8, Fig. 6.9,Fig. 6.10, Fig. 6.11,Fig. 6.12, and Fig. 6.13 shows the estimation processes at 9th-12 levels for L=1.31m. The tends of variance P at all the levels show that the estimations of the recursive least square method is convergent. The figure at 10th level indicates a smooth curve of the estimation, which results in the best estimate.

Fig. 6.14, 6.15 and 6.16 indicate that the estimation reaches the minimum $|\frac{\Delta L}{L}| =$ 10.5%, 7.9% and 2.6% at level 11, 10 and 10, respectively. The figures also show that after the estimation level at which the estimation error takes a minimum value, the value of $\tilde{e}$ and the estimation errors are increasing, due to lacking of the high

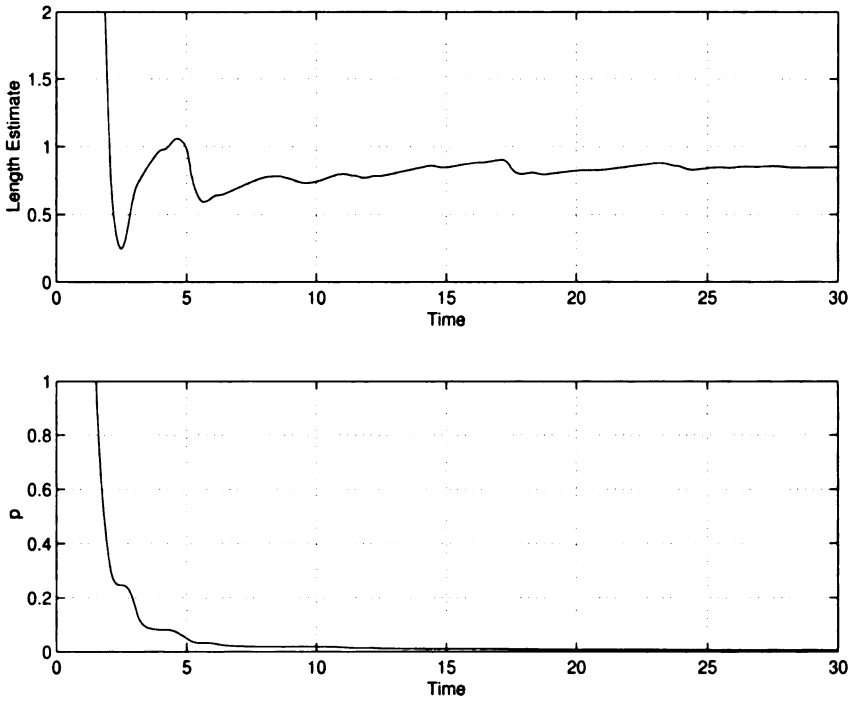


Figure 6.7: The length estimate and variance P at 10th levels for L=1.31m.

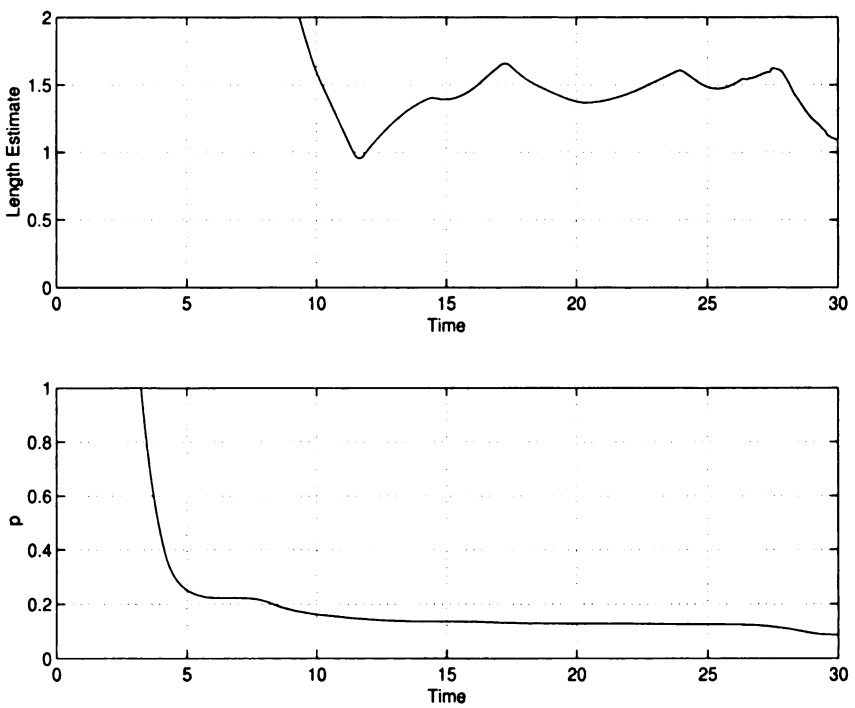


Figure 6.8: The length estimate and variance P at 11th levels for L=1.31m.

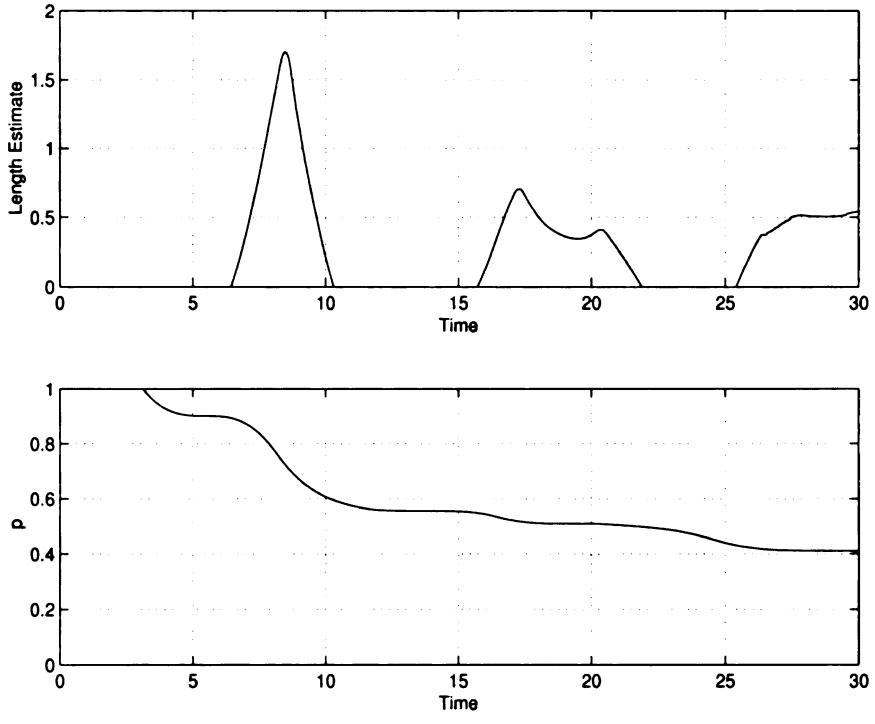


Figure 6.9: The length estimate and variance P at 12th levels for L=1.31m.

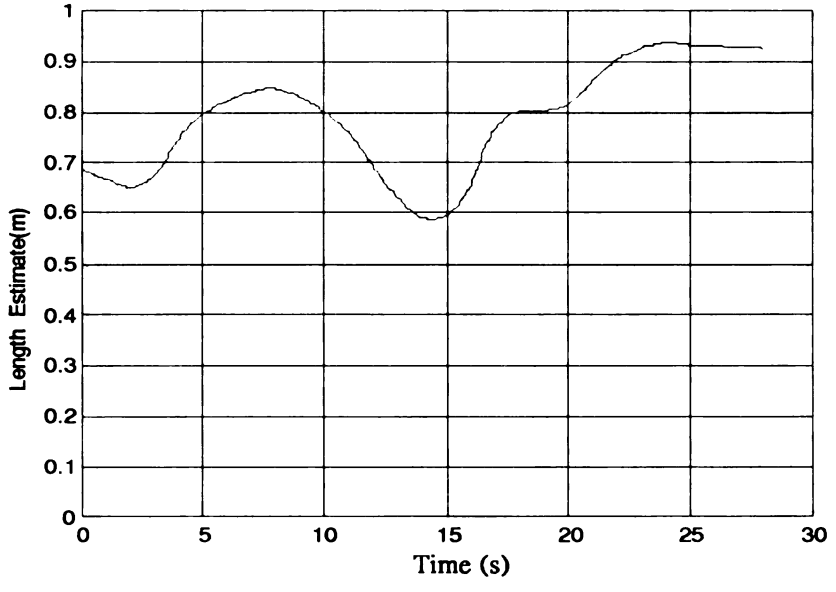


Figure 6.10: The length estimate and variance P at 9th levels for L=0.93m.

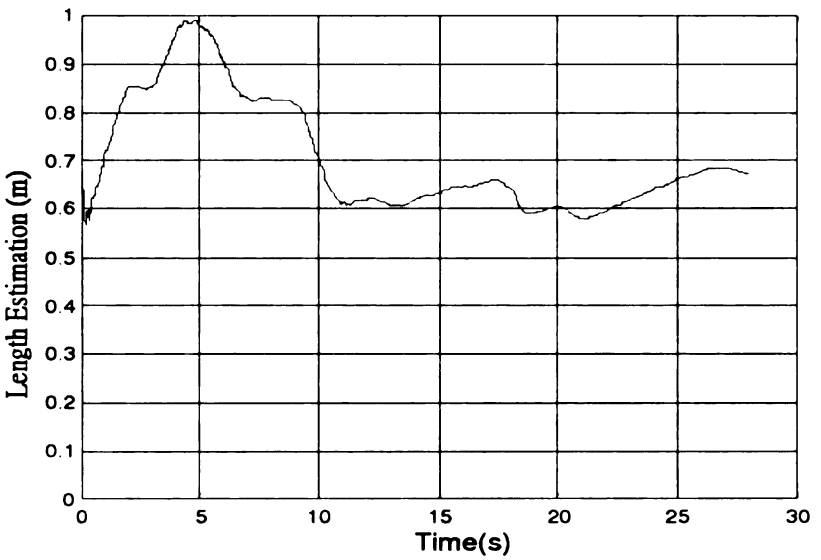


Figure 6.11: The length estimate and variance P at 10th levels for L=0.93m.

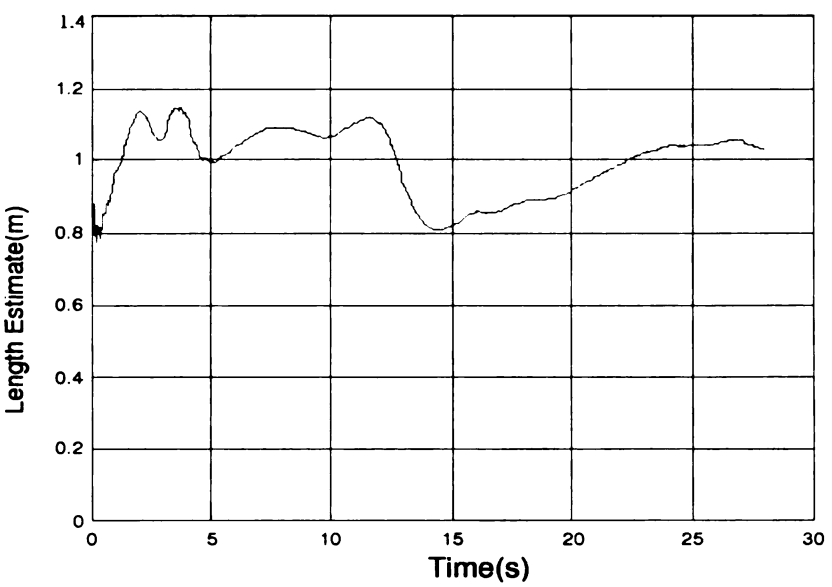


Figure 6.12: The length estimate and variance P at 11th levels for L=0.93m.

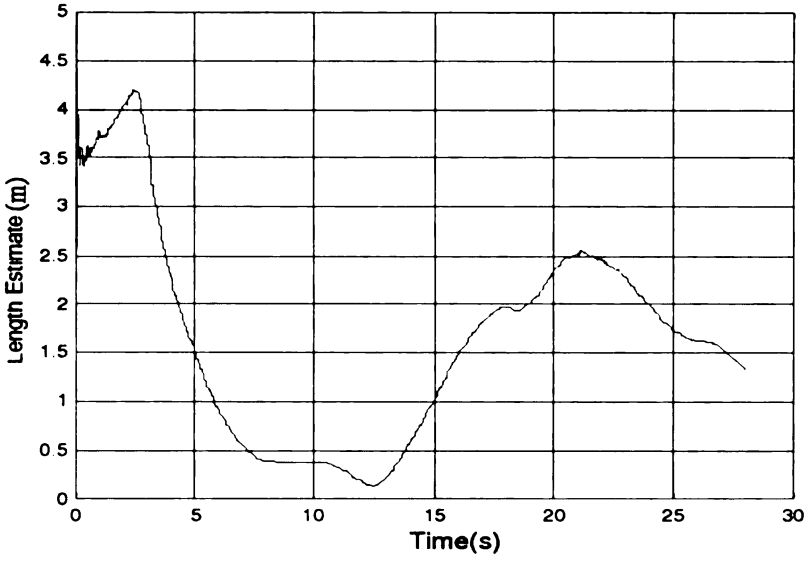


Figure 6.13: The length estimate and variance P at 12th levels for L=0.93m.

Length	LS		WBMI	
(m)	$\hat{L}$(m)	*error*	$\hat{L}(m)$	*error*
0.93	0.0290	-96%	1.0278	10.5%
1.14	0.1280	-89.3%	1.061	-7.0%
1.31	0.1213	-90.0%	1.415	7.9%
1.46	0.1577	-89%	1.50	2.6%

Table 6.2: Comparison of the Length Estimation Results.

frequency components of the true signal (serious distortion)at the further levels of low pass filtering.

The results from using the proposed method and traditional RLSM are compared in Table 6.2.

To estimate the kinematic length of a cart, the estimates by WBMI Algorithm, according to the proposed method, and the estimates by traditional LSM without preprocessing the raw data are used in model identification.

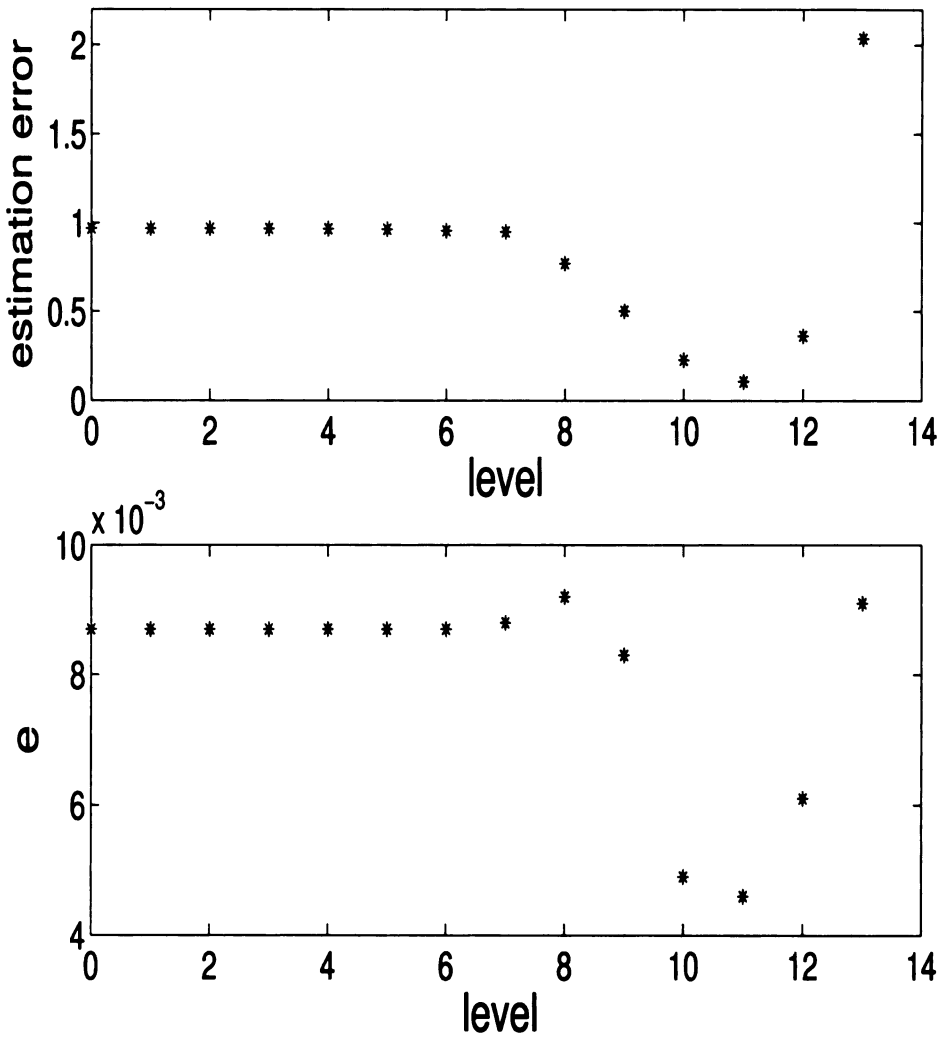


Figure 6.14: The results of $\tilde{e}$ and $|\frac{\Delta L}{L}|$ for L=0.93m.

6.3 Experimental Results for Hybrid Linguistic Control in Perceptive Frame

The proposed perceptive control model has been tested using a Mobile Manipulator-Phantom Joystick tele-operation System consisting of a Nomadic XR4000 mobile robot, a puma560 robot arm mounted on the mobile robot and a phantom joystick controller. as shown in Fig. 6.20. There are two PCs on the mobile platform for the control of the mobile robot and the Puma, respectively. Another PC is driving the Phantom joystick and communicating with the Puma controller through the Internet.

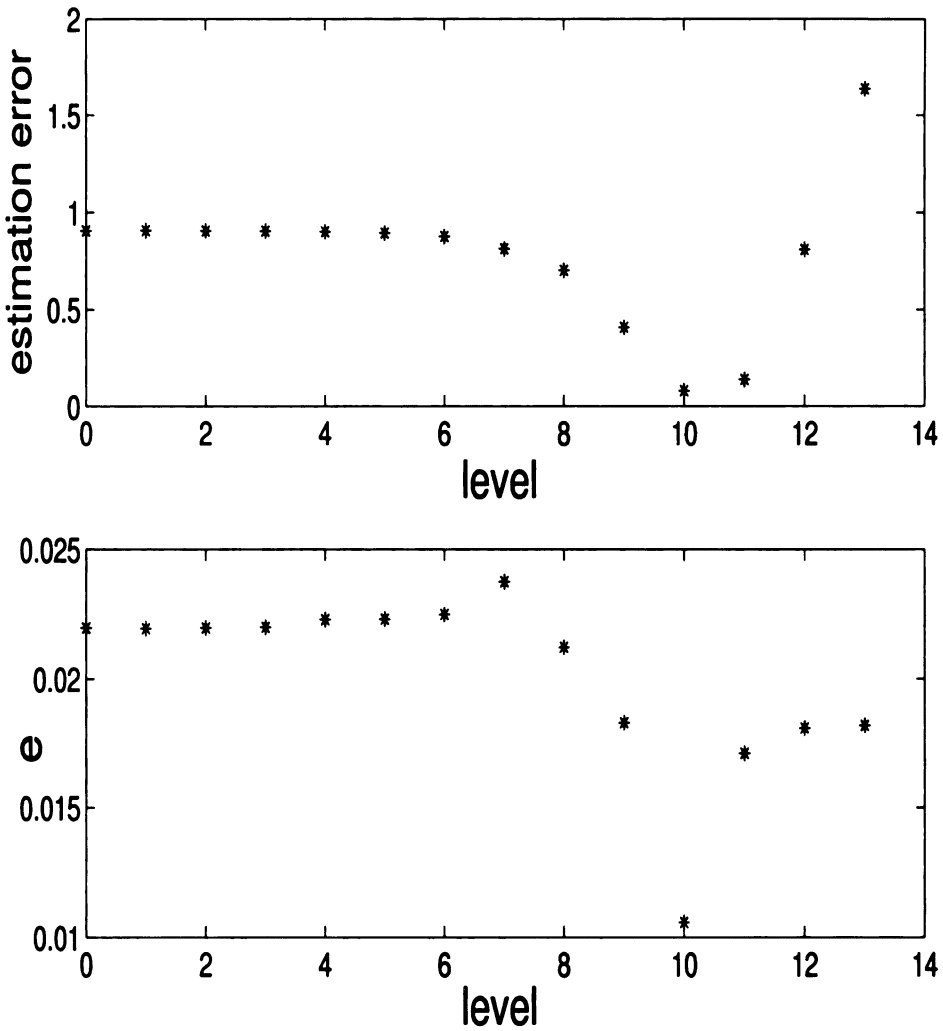


Figure 6.15: The results of $\tilde{e}$ and $|\frac{\Delta L}{L}|$ for L=1.31m.

The first task is designed as picking up an object followed by transporting it to a destination autonomously, i.e., without tele-operation. An unexpected obstacle blocks the mobile robot and prevents the continuous reference from evolving while the robot is executing the transporting task. In this implementation, environmental information is involved in the reference processing. The experimental results show that the proposed model is able to keep the perceptive reference evolving and trigger multiple events.

The experimental implementation is planned as a phase triggering multiple actions, a phase executing task, a phase modifying plan and a phase resuming task as

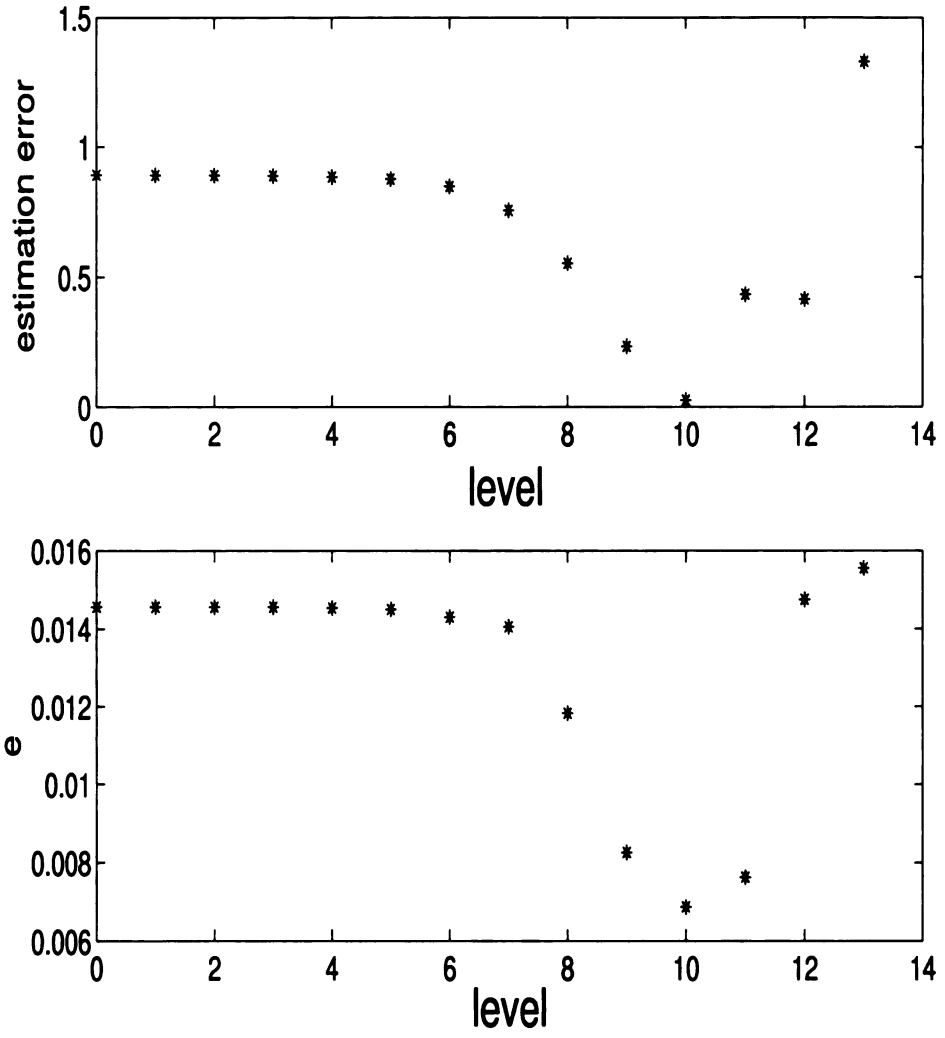


Figure 6.16: The results of $\tilde{e}$ and $|\frac{\Delta L}{L}|$ for L=1.46m.

mentioned in section 4.3.

As mentioned in the example, the experimental implementation is planned as a combined process including a phase triggering multiple actions, a phase executing task, a phase modifying plan and a phase resuming task.

Figure 7.11 shows the trajectory of the expirement result. The movement started from $P0(0.0, 0.0)$ and went straight to point $P1(710.0, 705.0)$, then picked up an object at this point. In the way of transporting the object to $P4(712.0, 2685.0)$, it modified the designed trajectory and made an obstacle avoidance at $P2(703.0, 865.0)$, from there, it went through a segment of an arc with a length of π. From point

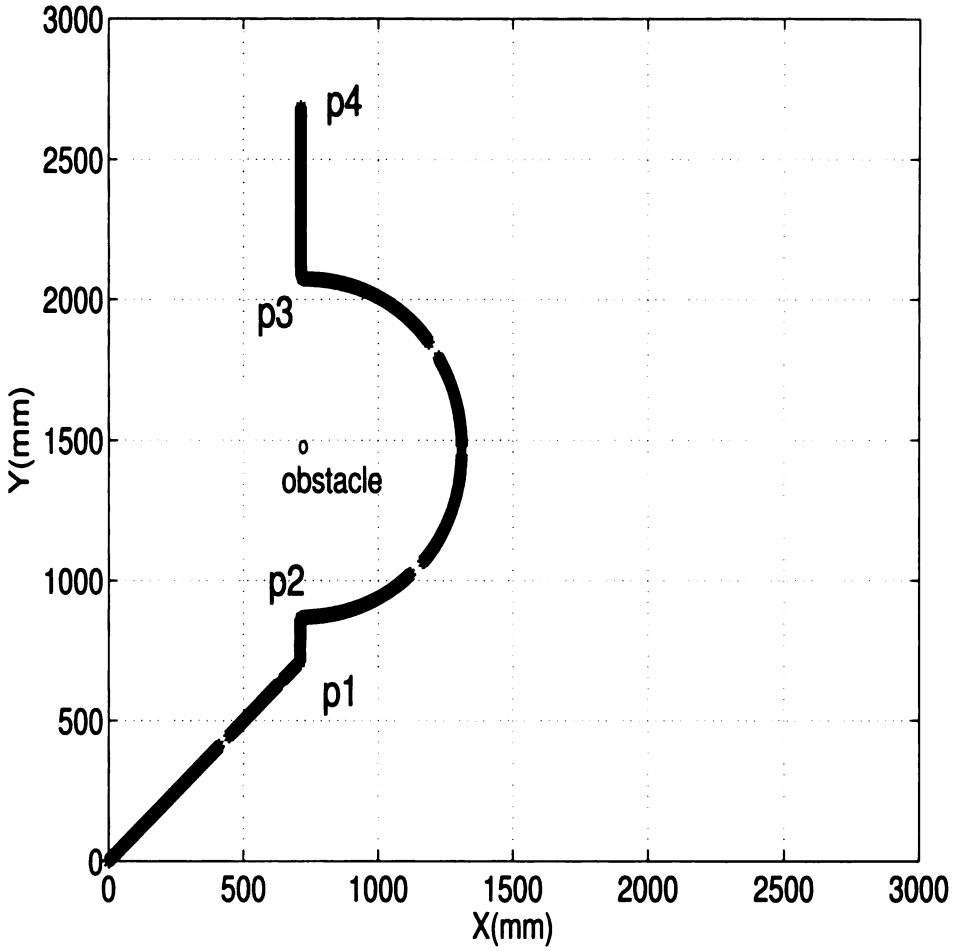


Figure 6.17: The moving trajectory in the experiment.

$P3(714.0, 2076)$ resumed the previously designed trajectory.

As designed, the action reference has triggered multiple events including next moving path segment and the end-effector actions at the point $P1$ and the point $P4$, the actions "close the gripper"at $P1$ and "open the gripper" at $P4$ are shown in Figure 6.19.

The second task is designed on a Phantom Joystick-Puma560 arm joint tele-operation to show unexpected event processing at the action level. As shown in Figure6.20, a phantom joystick is used to control a Puma 560 robot arm through the Internet. Phantom controller can measure and send out the position of the joystick in the phantom work space with a fixed rate of 10 Hz. The Puma arm controller

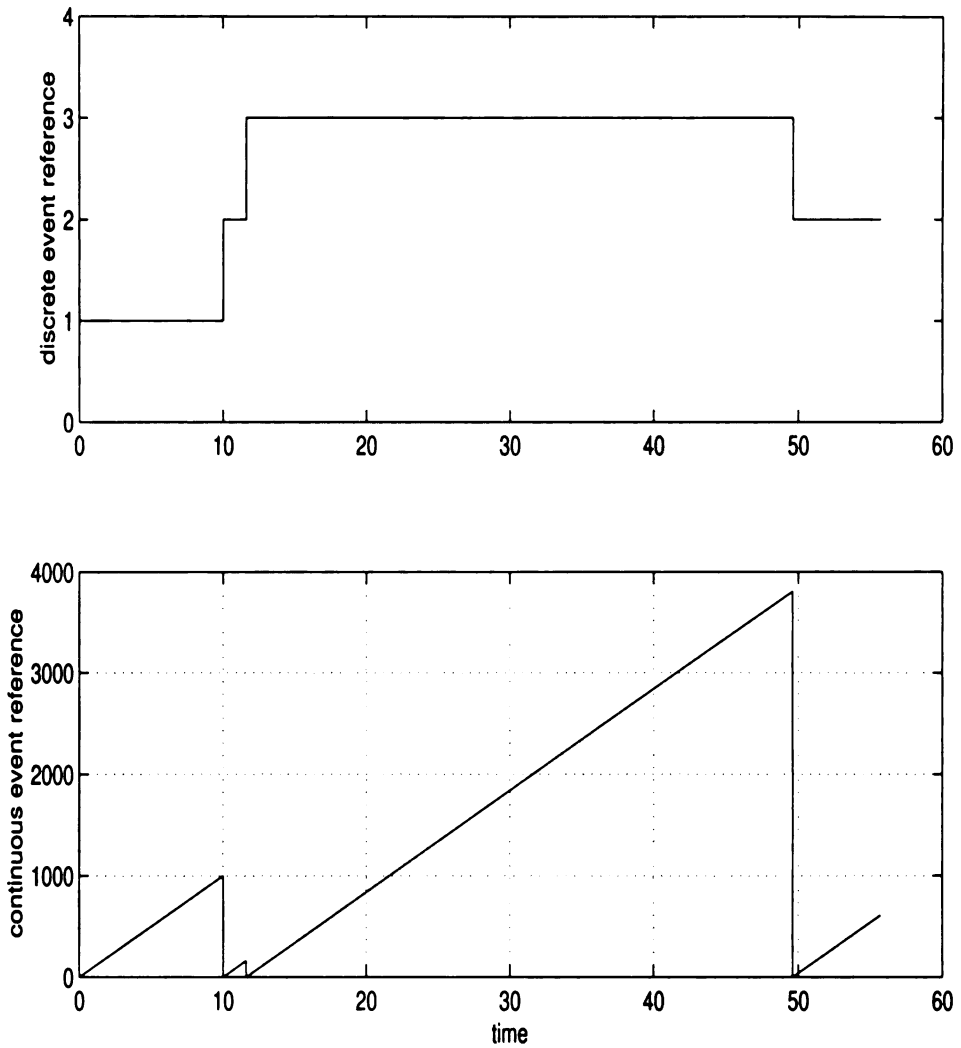


Figure 6.18: The moving reference in the experiment.

receives the information as motion commands and maps it to the puma arm work space. The control frequency of the arm controller is 500Hz. Therefore, the arm runs 50 control cycles for one phantom command. The task is to implement position tracking from Phantom space to Puma arm space in real-time. This experiment is designed for processing the expected events caused by the time delay of the Internet communication between the joystick and controller. If the phantom controller cannot receive the next position command within 50 cycles of the current command, it is an unexpected event which can block the discrete perceptive reference.

Figures 6.21 6.22 6.23 show the Phantom-Puma position tracking control result on

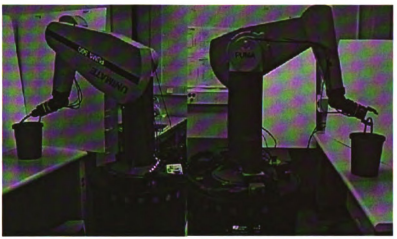


Figure 6.19: The actions triggered by the action level reference.

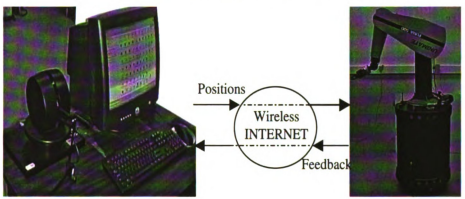


Figure 6.20: The phantom joystick tele-operation.

z axis. Fig A illustrates the trajectory from phantom position data. Fig.C shows that the Puma position tracking is smooth with unexpected event processing. As shown in D, it can be found that the unexpected events (delayed commands) occurred during the task execution. Corresponding to Fig D, Fig. B shows that the velocity control signal is continuously evolving, i.e., the proposed hybrid reference cannot be blocked at the occurrence of the unexpected events that may block the evolution of the single discrete perceptive reference.

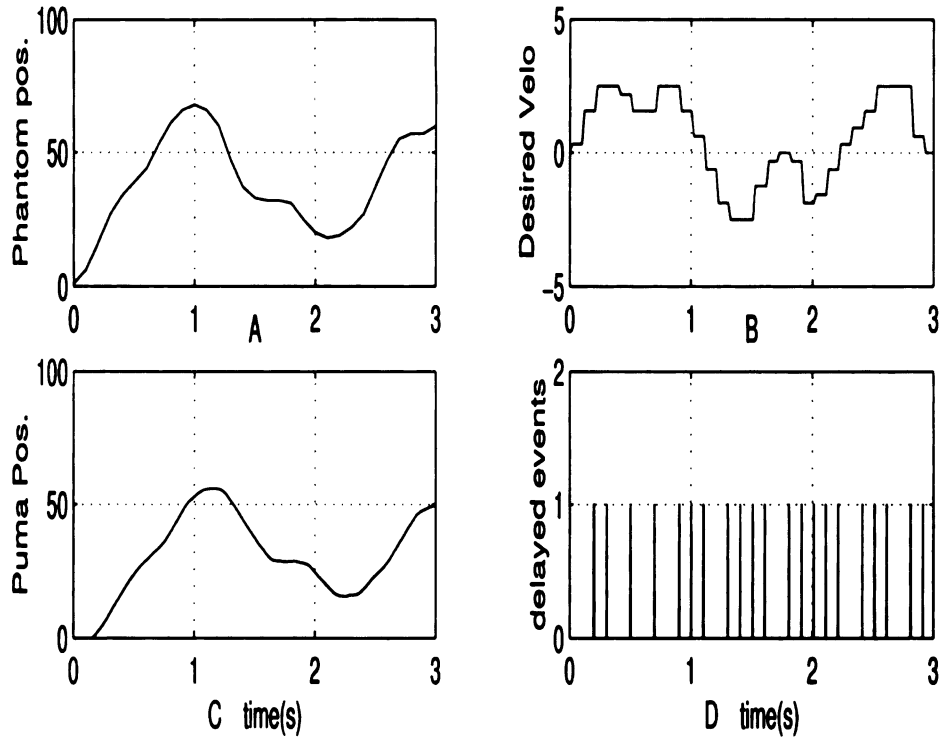


Figure 6.21: Unexpected Events in Phantom Manipulation along x-axis

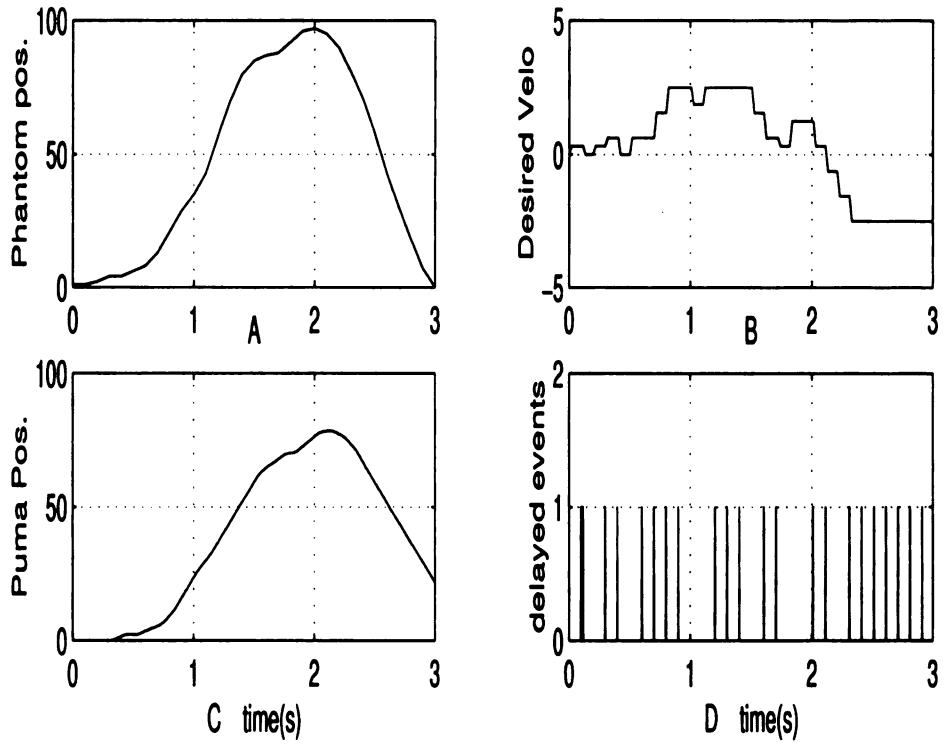


Figure 6.22: Unexpected Events in Phantom Manipulation along y-axis

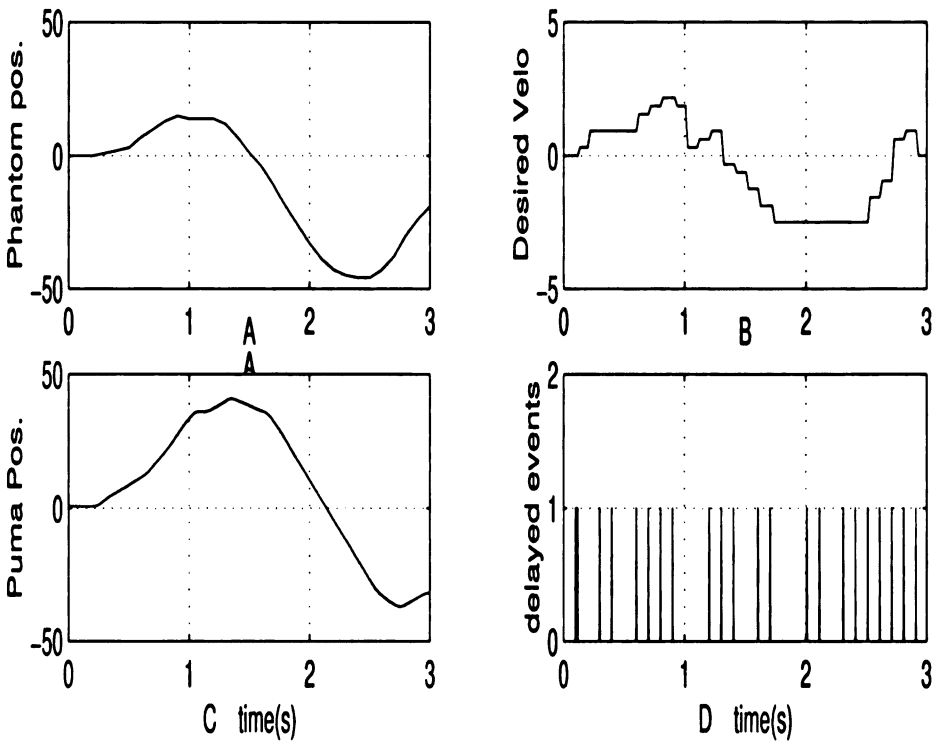


Figure 6.23: Unexpected Events in Phantom Manipulation along z-axis

CHAPTER 7

PERCEPTIVE CONTROL SYSTEM DESIGN AND VLSI IMPLEMENTATION

Nowadays, real-time (or embedded) systems [12] have been attracting people's attention. Roughly speaking, a real-time system is required to react to stimuli from the environment (including the passage of physical time) within time intervals dictated by the environment. In other words, real-time systems must respond to real-world events. Naturally, this class of systems can be implemented in the perceptive frame.

The perceptive frame control method is aimed to solve the problems in Real-Time Control System design. In this chapter, a real-time programming language is proposed for designing real-time control systems. Two important mechanisms, event triggering and sampling are discussed in this chapter.

7.1 Real-time System in Perceptive Frame

7.1.1 Input-Reference-Output Mapping in the Perceptive Frame

The task description space can be formulated as $U = \{T_{TRi}T_{ARj}I_{tk}\}ijk$, a combination of segment in the task space. Real-time control requires the system to respond to the events from sensor measurements, whenever they happen. Thus, events are possible at any point on the time axis.

The system can be thought of as a mapping: $U \times E \rightarrow Output$, where the *Output* is the set of real-time related actions, including event triggering and sampling.

As studied above, every hybrid perceptive reference trajectory $\{S_{TRi}S_{ARj}I_{sk}\}ijk$ is an ordered set with an order "$\preceq$". The trajectory is a combination of the ordered closed intervals. Every hybrid perceptive reference trajectory can be mapped onto a time interval linearly. This provides the feasibility of performing real-time operations

in the perceptive frame, i.e., based on the hybrid perceptive references.

7.1.2 Perceptive Reference Based Language Description Regarding Real-Time Control System Design

To a large extent, there exist a variety of tasks and actions in real-time control systems. The objective of the system design is to implement the tasks and the actions in the perceptive frame with the architecture shown in Figure 4.2.

In order to solve the design problems efficiently, a real-time language is desirable to describe the relation between the commands (high level and low level), the perceptive references and the real-time related outputs. The introduction of the perceptive references into the language enables the system to deal with both the continuous and the discrete events.

7.1.3 Finite Automata Approach for Input-Reference-Output Mapping

Automaton is a mathematical model to depict the logic behaviors of a system. In the perceptive frame, the tasks and actions are required to be executed sequentially. Hence all the discrete transitions can be modeled by finite automata. The task information can be decomposed into different lower levels to be expressed based on its semantic description.

$$q_j = \delta(q_i, \sigma_j, X), \delta_j \in \Sigma \tag{7.1}$$

$$O_k = \eta(q_i, \sigma_i, X) \tag{7.2}$$

7.2 VLSI Implementation-Oriented Controller Design

7.2.1 Abstractions of real-time perceptive controller

The control implementation can be represented at the different levels of abstraction: entity abstraction, procedure abstraction, and hierarchy abstraction.

Abstraction of the controller entity

In the perceptive frame, the hybrid control systems are the combinations of control primitives. Each control primitive can be treated as a control entity. A control entity is a controller that can execute a control command at certain level. The control entity has the specifications to describe the inputs, the outputs, and life time of the entity. The lifetime of the entity in the perceptive frame is a segment of the perceptive reference, it marked the starting point and the end point for the entity to be activated. In the implementation, a task can be composed of several entities at lower level.

In the system shown in Figure 4.2, a control entity can be a controller for executing a task consisting of action entities.

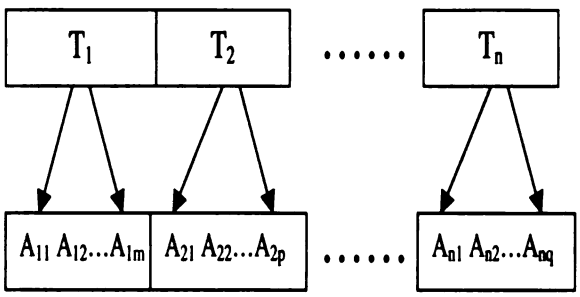


Figure 7.1: Abstract of the structure.

The control entity can be described as an object in pseudo-code.

Entity {
inputs,
outputs,
active period in perceptive frame,
entities at lower levels,
}

Abstraction of the logic control structure: 1. Sequential Structure.

The operations of the automata can show the logic control structure of the automata. For a typical perceptive automaton, The transition function is a branch control structure in sense of logic.

$$q_j = \delta(q_i, \sigma_j), \delta_j \in \Sigma \tag{7.3}$$

The perceptive referenced output function has loop structure.

$$O_k = \eta(q_i, \sigma_i) \tag{7.4}$$

The figure 7.2 indicates the logic structure of an automaton, the basic sequential logic structures in the implementation are branches and loops, i.e., the selective structure and the repetitive structure.

2. Parallel Structure.

Figure 4.2 shows that the reference generation and the task execution are working concurrently. Furthermore, the automaton have multiple actions triggered by the task, for example one task can trigger the actions performed by the mobile robot and the robot arm at the same time. There are more than one process running in a processor.

Abstraction of the Functional Hierarchy:

The Figure 4.2, it can be seen that the task scheduler, the action planning and the motion planner are sequentially chained together. The input of one automaton is output of the higher level automaton. Hierarchy.

In order to implement the design, the above abstractions discovered the structure of the system in three aspects, i.e., the controller entities, the functional structure, and the control structure. The design procedure is shown in Figure 7.3

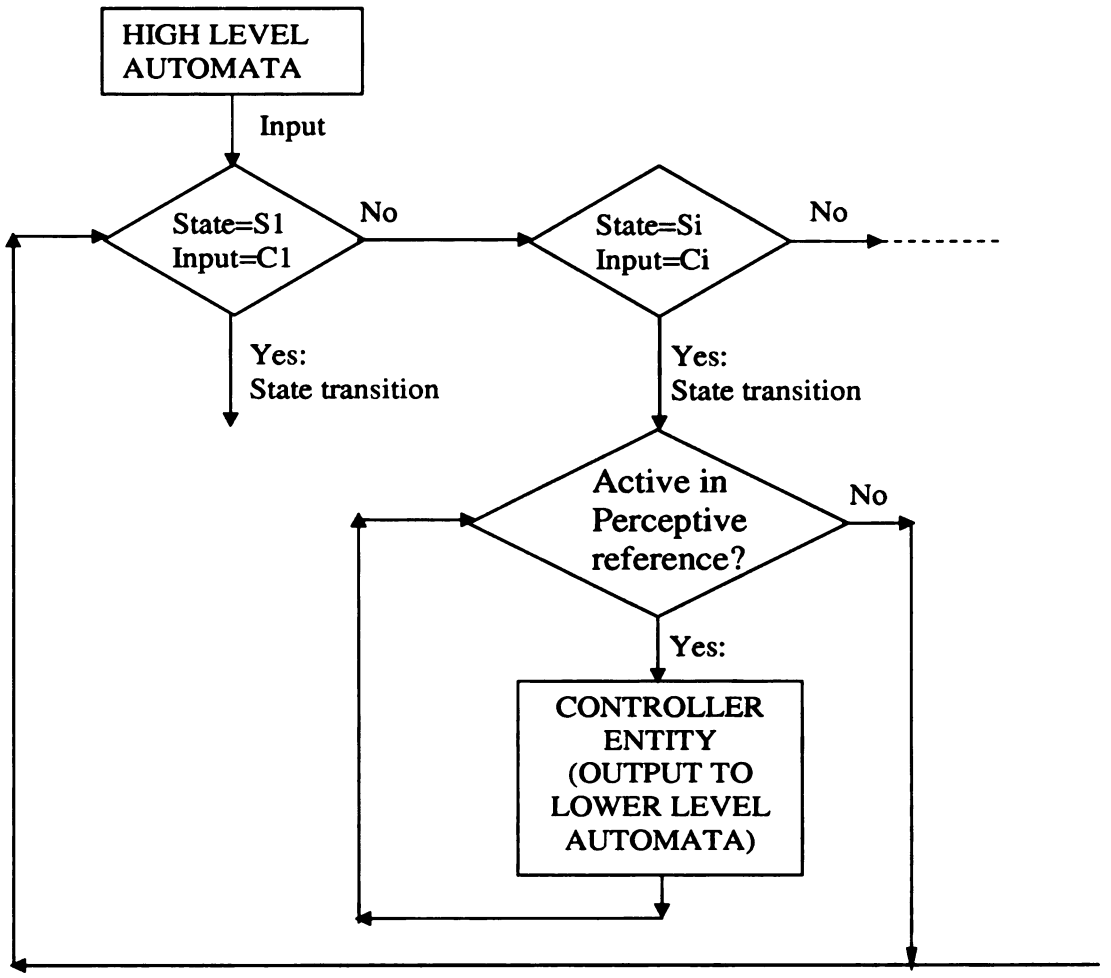


Figure 7.2: Structure of the Implementation

Event Triggering

Timing is crucial in any real-time system. It is required to have real-time control facilities, including specifying times at which actions are to be performed and completed, and responding to situations where all the timing requirements cannot be met. The time triggering plays a key role in real-time systems.

The unit step function:

$$u(t - t_0) = \begin{cases} 0 & t < t_0 \\ 1 & t \geq t_0 \end{cases} \tag{7.5}$$

is a simple example of time triggering.

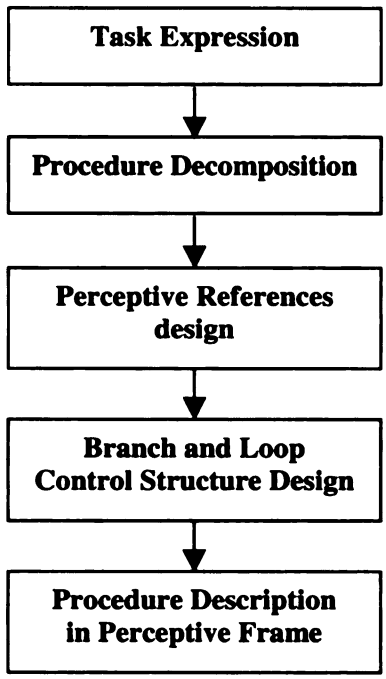


Figure 7.3: Block Diagram of the Implementation Procedure.

In order to make the real-time system purely independent from time t, in the perceptive frame, the perceptive reference replaces the time as the system reference. Thus, the problem of event triggering is generally unavoidable in the real-time systems.

Sampling System One of the typical problems in real-time systems is sampling data. Information carried by an analog signal can be represented in a digital form by a sequence of its instantaneous values measured at discrete time instances. These signal readings are usually considered as signal sample values and the process of taking them is referred to as sampling. The instances at which the samples are obtained form a stream of uniform events, which can be depicted graphically as a sampling point process. Characteristic features of the sampled signals to a large extent depend on the patterns of the point processes generated and used for sampling [48]. When sampling is mentioned in the context of Digital Signal Processing(DSP), usually it is assumed that the sampling considered is deterministic and periodic. The model of sampling according to which signal samples are separated by time intervals with a constant

and known duration is the most popular one. This is readily comprehensible because such a sampling approach appears to be the most natural and obvious, as shown in the following equation

$$x[n] = x_c(nT), -\infty < n < \infty \tag{7.6}$$

Where x_c is a continuous-time signal, $x[n]$ is a sequence of samples obtained from x_c, T is the sampling period and its reciprocal is the sampling frequency.

For a control method in the perceptive frame, time t is not used as the common reference. It is desirable to design a sampling mechanism with the perceptive references. Roughly speaking, the sampling in the perceptive frame can be explained as a process to take a sequence of points of a continuous signal with the marks in the perceptive references. The sampling period is not characterized by time, but perceptive reference. In other words, the density of the sampling on a signal depends on the environmental sensing. If the sensor information changes rapidly, the signal is sampled quite frequently. The sequence of the samples can be depicted as

$$x[n] = x_c(s_n^h), -\infty < n < \infty \tag{7.7}$$

where the s_n^h represents the instances of the hybrid perceptive reference at which the samples are obtained.

7.2.2 Synthesizability of Perceptive Controller

In general, the term "synthesis" is referred to as the automated transformation of RT level descriptions into gate level representations, or other more detailed descriptions [70]. This transformation is mainly influenced by the set of basic cells that are available in the target technology. While simple operations like comparisons and either/or decisions are easily mapped to Boolean functions, more complex constructs

like mathematical operators are mapped to a tool specific macro cell library first, for example, boolean network or finite state machine.

Functionally, the discrete controllers can be described as finite automata, which are the devices to accept the strings in a formal language. For a finite automaton, if we can find a set of Boolean Functions, or a Boolean Network, set B_f connected to a finite state machine, consisting of flip-flop components, $M(Q, I)$, Q is the state set of the finite state machine, I is the input language. Then, the finite automaton is synthesizable, tuple $(B_f, M(Q, I))$ is an implementation of the finite automaton.

Theorem : *Given that a finite automaton is synthesizable, there exists a tuple* $(B_f, M(Q, I))$, *which is an implementation of the finite automaton. Based on the discrete transition of the finite automaton,*

(1)Perceptive automaton is synthesizable.

(2)Hybrid automaton is synthesizable.

(3)Furthermore, the multiple automata system, shown in Figure 4.2, i.e., connected by task/action description language and reference language, is synthesizable.

Proof:

(1)The perceptive automaton has two linguistic inputs, namely the task inputs formed from Σ and the reference input formed from Σ_R. The reference inputs change the linguistic output O, $O_k = \eta(q_i, \sigma_k)$, which can be described by Boolean logic equations. Comparing to the implementation $(B_f, M(Q, I))$ for the fundamental finite automaton, the Boolean function B_f of perceptive automaton depends also on the reference input I_R, The $B_f(I_R)$ is the Boolean function set for the new automaton. $(B_f(I_R), M(Q, I))$ is an implementation for the perceptive automaton. M_e, therefore, is synthesizable.

(2)The state transition function $q_j = \delta(q_i, \sigma_j, X), \delta_j \in \Sigma$ and the output function $O_k = \eta(q_i, \sigma_i, X)$ show the new features of the hybrid automaton. Hybrid automaton has continuous variables, in digital systems, the variables can be discretized and

represented by fixed point number, or, for the sake of simplification, by integers N. Those discretized variables are the operands for specific arithmetic operation, i.e., for specific discrete state q_i. In hybrid automaton, arithmetic operations can be performed by Boolean logic operation added to the Boolean implementation B_fh. Hence, $(B_{fh}, M(Q, I))$ is an implementation of the hybrid automaton. The automaton is synthesizable.

(3)The integrated system shown in figure 4.2 is a set of connected automata, which are perceptive automata, or hybrid automata or both. According to the first and the second items of the theorem, each automaton is synthesizable, i.e., each automaton has a tuple $(B_{fi}, M_i), i = 1...n$ as its implementation. The input and the output of the single automaton can be treated as strings in the formal languages with discretized valuables, they can be encoded with limited number of bits and expressed by Boolean functions.

(a) For two automata connected by perceptive reference language L_R. Automaton 1 has the output as automaton 2's reference input. The implementations (B_{f1}, M_1) and (B_{f2}, M_2) exist for two automata, respectively. The input L_R effect only the logic operation, no state transition. In order to involve the implementation of 1 to 2, we substitute the tuple $(B_{f1}, M1)$ into 2, the implementation for combined automaton is $(B_{f2}(B_{f1}, M_1)), M_2)$.

(b) For two automata connected by task language input L_T. The automaton 1 has the output as automaton 2's task input. The task input effect only on the state transition. Therefore, the implementation for 1 and 2 is $(B_{f2}, M2(B_{f1}, M_1))$. The combined system is synthesizable.

For the sake of induction, it can be assumed that for N automata connected by the task and reference inputs are synthesizable. Then an implementation tuple (B_{fn}, M_n) can be found. When we have $N + 1_{th}$ automaton, which is synthesized as (B_{fn+1}, M_{n+1}), under the task inputs and the reference inputs, similar to (a) and (b), there exists an

implementation (B_f, M). The integrated system is synthesizable.$\triangle$

7.3 Hardware-Software Co-design and Hybrid Simulation

7.3.1 Hardware-Software Co-design in Perceptive Frame

Hardware-Software Co-design has been attractive topic in realtime system design. The hardware is the key part of the embedded system that can improve the real time processing performance, the software can provide more flexible characteristics of the high level task description. The perceptive frame makes the system more flexible on software/hardware design, because of its modularized architecture. Consider in task execution, the system can have different partitioning as shown in figure 7.4. We can build micro-instructions based one different partitions, the micro instructions are implemented by hardware, and based on which, the software can be programed with these instructions to utilize the hardware resource at the best.

Implementation Oriented Language(IOL)

The abstractions above show that the implementation of the real-time control system can have a procedural description, a functional description, and an implementation-oriented description. Based on the specific semantics, the programming language can de designed with syntax. A convenient way to convert the system design to implementation is mapping the design to a hardware design language, for example, VHDL. In Chapter 4, we have introduced the hybrid formal languages in the perceptive frame, including L_T^h, L_A^h, L_{RT}^h, and L_{RA}^h. The languages describe the tasks, the actions, the task level reference, and the action level reference. A programming language can be built based on the description languages.

RTL languages are widely used to express complex digital systems concepts. For hybrid model in the perceptive frame, Figure 7.3 give the procedure for system design.

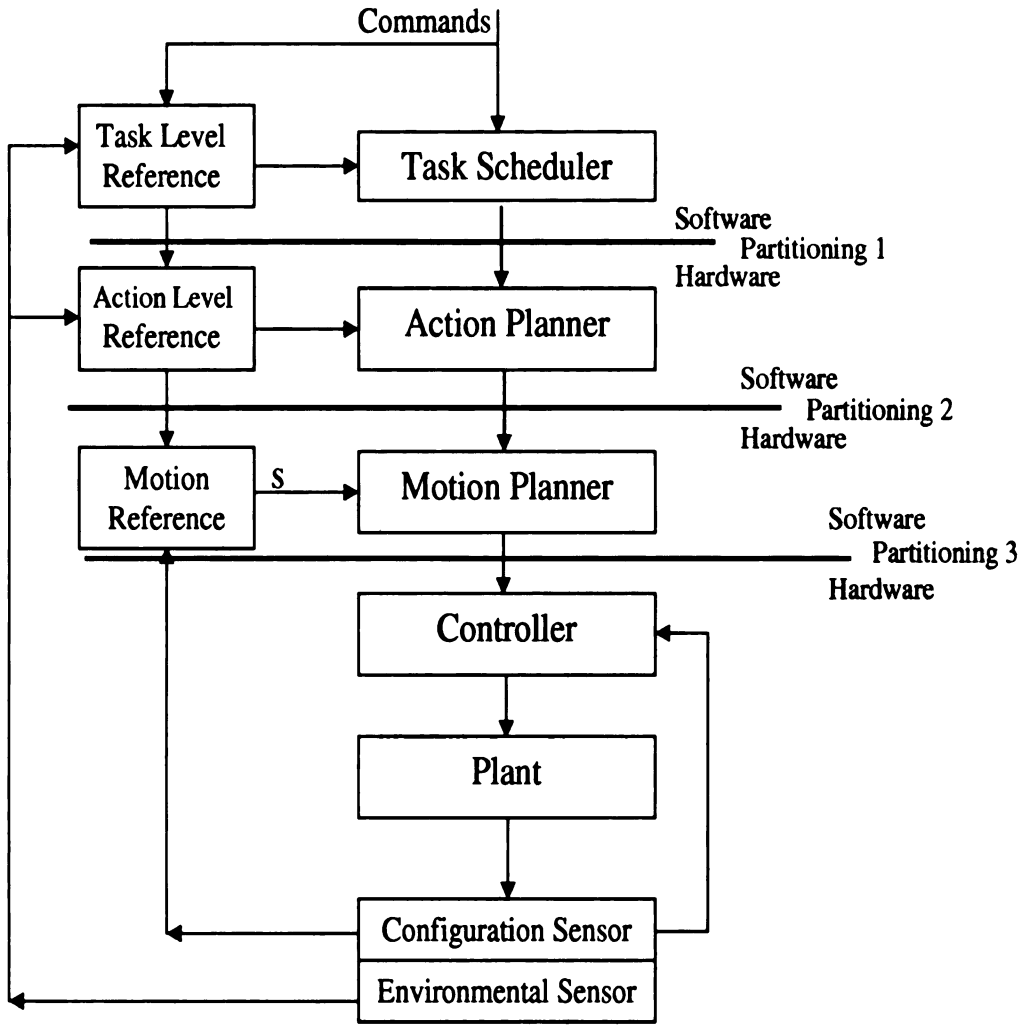


Figure 7.4: Partitionings for Hardware/Software Codesign.

Following this procedure, one can obtain the linguistic expression based on languages L_T^h, L_A^h, L_{RT}^h, and L_{RA}^h to desctibe the Register Transfer Level Objects in perceptive frame. The words in these languages are used to express the system behaviors. A sequence of tasks and actions can be expressed by the IOL,

$T_1(S_{T1b}, S_{T1f})\{A_{11}(S_{A11b}, S_{A11f}); A_{12}(S_{A12b}, S_{A12f})\};$

$T_2(S_{T2b}, S_{T2f})\{A_{21}(S_{A21b}, S_{A21f}...)\};...$

where T_1 and T_2 are the elements and control entities in L_T^h, $A_{11}A_{12}...$ are in included in language L_A^h. The perceptive reference values, which follow the task and action words, denote the active period for the specific entity in perceptive frame.

The semicolon indicates the sequential process, the parallel processing is expressed by the perceptive references. In the above expression, (S_{T1b}, S_{T1f}) is defined as the active period for entity T_1, in other words, task T_1 is started and ended by perceptive reference values S_{T1b} and S_{T1f}, respectively. The structure of the expression can easily transfer the control system architecture into the general description language with modularized description, such as Verilog.

The syntax can be described by the following parse trees for operator ";" and operator "(,)", respectively.

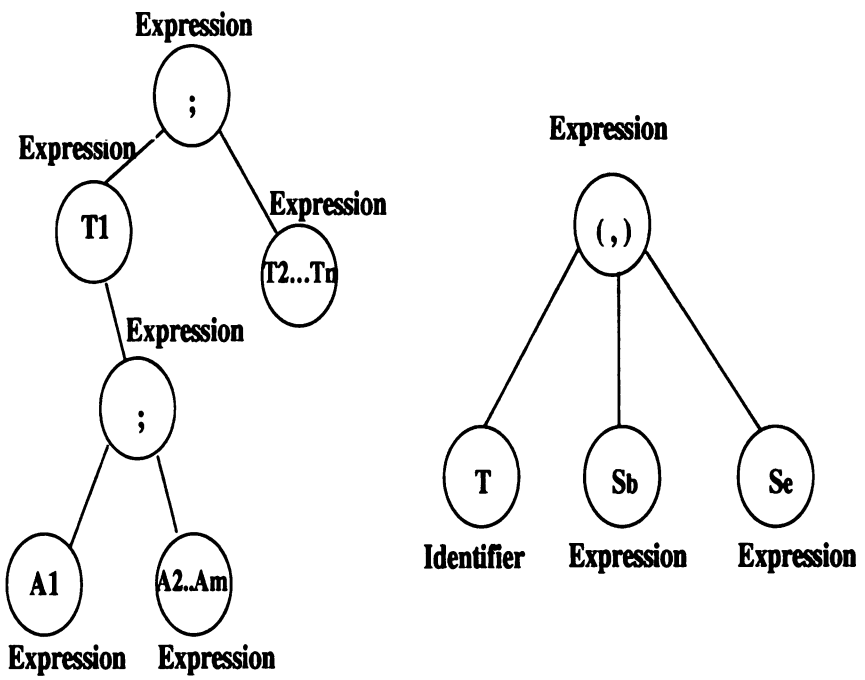


Figure 7.5: Parse Trees for Implementation Oriented Language.

Based on the parse trees, the high level behavior of the entire integrated automata can be constructed by the syntax. The semantics of the perceptive controllers can be performed by the Implementation Oriented Language.

An Example on IOL Programming

VHDL is a language used to express complex digital systems concepts. Figure 7.3 give the procedure for system design. Following this procedure, one can obtain the linguistic expression based on languages L_T^h, L_A^h, L_{RT}^h, and L_{RA}^h. The words in these languages are used to express the system behaviors. A sequence of tasks and actions can be expressed by the IOL (language),

$T_1(S_{T1b}, S_{T1f})\{A_{11}(S_{A11b}, S_{A11f})A_{12}(S_{A12b}, S_{A12f})\}$

$T_2(S_{T2b}, S_{T2f})\{A_{21}(S_{A21b}, S_{A21f})\}$,

where T_1 and T_2 are the elements in L_T^h, $A_{11}A_{12}...$ are in included in language L_A^h. The perceptive reference values, which follow the task and action words, denote the active period for the specific entity in perceptive frame. For example, (S_{T1b}, S_{T1f}) is defined as the active period for entity T_1, in other words, task T_1 is started and ended by perceptive reference values S_{T1b} and S_{T1f}, respectively.

VHDL is a language used to describe complex digital systems. In order to associate the IOL perceptive language with the hardware implementation, it is clear that we need to build a map from the IOL perceptive language to VHDL.

The program involves the synchronization mechanism and hierarchical structure of the entities. First, the tasks T_1 and T_2 are executed sequentially, while the actions may be executed in parallel. In this case, the actions are synchronized by the perceptive reference. Second, the actions are included in tasks, which belong to higher category of entity. Based on the abstraction of the controller entity, the entity in IOL consists of the inputs, outputs, active period for the entity as a feature, and other entities.

The sequential structure is:

```
P1: process:
Begin
    T1,
    T2,
End process;
```

Considering the perceptive referenced system, the signal mechanism in VHDL provide the synchronization, the perceptive reference can be expressed as a trigger. Thus, The controller entity can be described as:

```
Architecture T1 of Task is :
    signal TRIGER: perceptive reference
Begin
    process
    wait on TRIGGER;
    A11,
    A12,
    end process;
end T1;
```

The *TRIGGER* is the perceptive reference.

In addition to the task entitis and action entities, as the lowest level entity, the low level controller can be programmed in VHDL by its mathematic description. Usually, it is included in a library as a module.

7.3.2 Hybrid Simulation

The proposed controller design method has been tested using Verilog HDL and Cadence Design System. The functions of the controller can be performed by a Verilog HDL program, which is semantically equivalent to the hybrid perceptive model. Cadence design system is used to synthesize the Verilog program and simulate the given example. The simulation system consists of a controller simulator, a simulator for the continuous dynamics of the plant and a perceptive reference simulator. In order to simulate the hybrid properties of the system, TestBuilder, a C++ testbench class library, is used in the simulation system. By using Test Builder, the plant dynamics, consisting of Ordinary Differential Equations(ODE), can be simulated by a C++

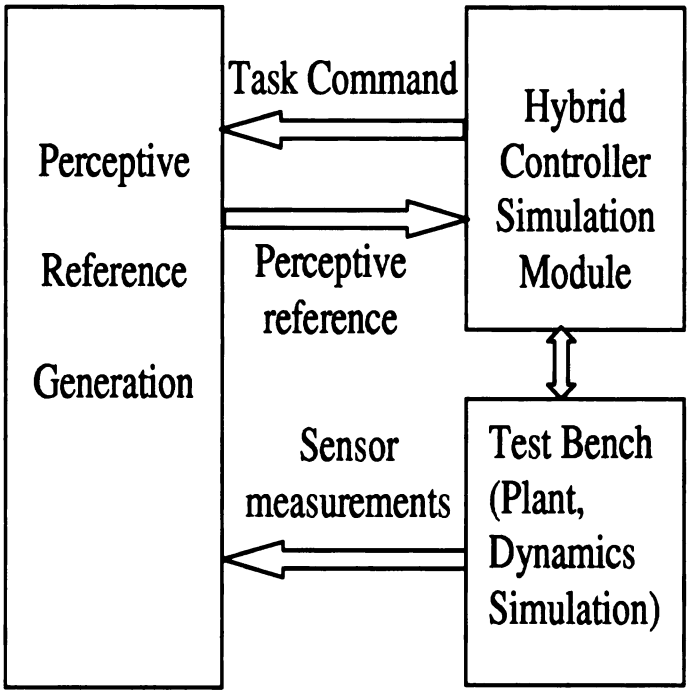


Figure 7.6: The Scheme of the Hybrid Simulation System.

program. As a testbench, the dynamics in C++ can communicate with HDL simulator in Cadence. Based on the velocity and position commands from the controller, the dynamics can give out the states of the system, including position, velocity, and sensor measurements. The modules are implemented independently from each other, they are connected only with the linguistic commands and the perceptive references.

Implementation of Arithmetic Operations

For a controllers, most frequently used operations are arithmetic operations. To reduce the scale of the circuit, we use fixed-point number representation for implementation the control algorithm. For a signed real number, the integer part is 11-bit long, the decimal part is 12-bit long. The arithmetic unit provided by Verilog is implementated with unsigned division and signed multiplication.

To keep the accuracy of division and multiplication in the controller, a new arithmetic unit is developed, as shown in Figure 7.7, the division and the multiplication

can be done by Verilog with the desired accuracy.

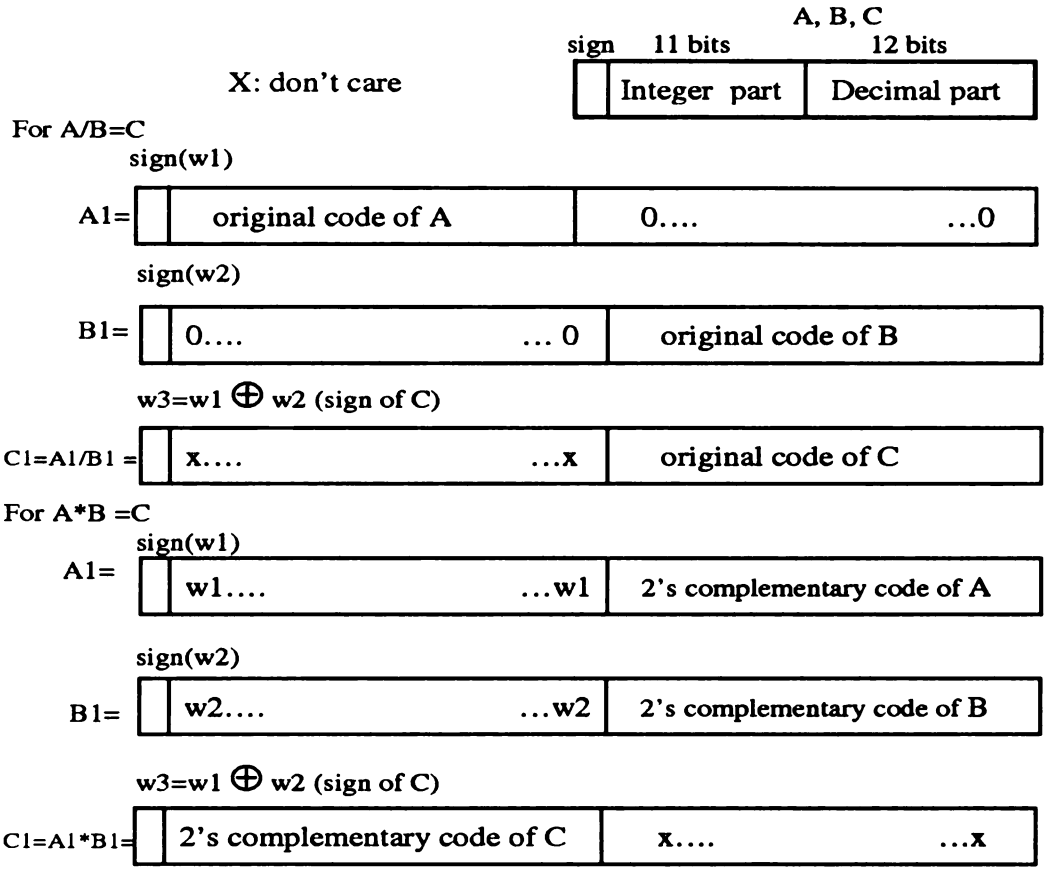


Figure 7.7: Arithmetic Operations.

Dynamics Model in Testbuilder

The dynamics model in Testbuilder is an ODE model. In perceptive frame, a mobile robot model can be described as:

$$\frac{dw}{ds} = 2a \tag{7.8}$$

$$\frac{da}{ds} = u \tag{7.9}$$

For a robot arm, dynamics model is given by

$$D(q)\ddot{q} + C(q,\dot{q}) + G(q) + J_h^T(q)f_e = \tau \tag{7.10}$$

7.3.3 Simulation results

The first task of the simulation is to generate the control signal for the robotic system to go through a desired path with two straight lines. The robot travels toward the wall, before reaching the wall, the robot turns and travels along the wall to go to the destination. In the perceptive frame, the action A_1 and A_3 action command will be triggered by the perceptive reference. The robot will move along X direction for 3 units, then travel along Y direction for 4 units.

As described below, the task, denoted as $T_1(3,4)$, means move to point(3,4). Assuming the current position is (0,0), the task can be decomposed as two actions, namely, A_1, which means traveling from current position to (3,0), and action A_2 for moving from (3,0) to (3,4). The perceptive references will be the signals to indicate the distance the robot traveled and to trigger the next action, while the current action is finished.

For a typical perceptive hybrid automaton, the action planner has two inputs, the command input and the reference input, and an output. Figure 7.8 illustrates the schematic description synthesized by Cadence Design System. The command inputs are $Task$, $Task_x$ and $Task_y$, the reference input is $ActionRef$. F is the output to the motion planner as its command input. Therefore, the perceptive/hybrid automaton is synthesizable. Figure 7.9 illustrates the implementation of Action Reference in the given case. According to (4.22),(4.23), the wall signal is generated from the sensor measurements. The wall signal indicades that the wall is detected and can make the action reference to switch from 1 to 2. The low level controller is a digital circuit to perform the low level control algorithm, it is the interface between the continuous dynamics and the higher level discrete planners.

The integrated automata model shown in Figure 4.2 have been tested using Cadence. As proved before, the entire system is synthesizable. Based on the synthesized description, the simulation has been performed. Figure 7.10 is the signal wave form of

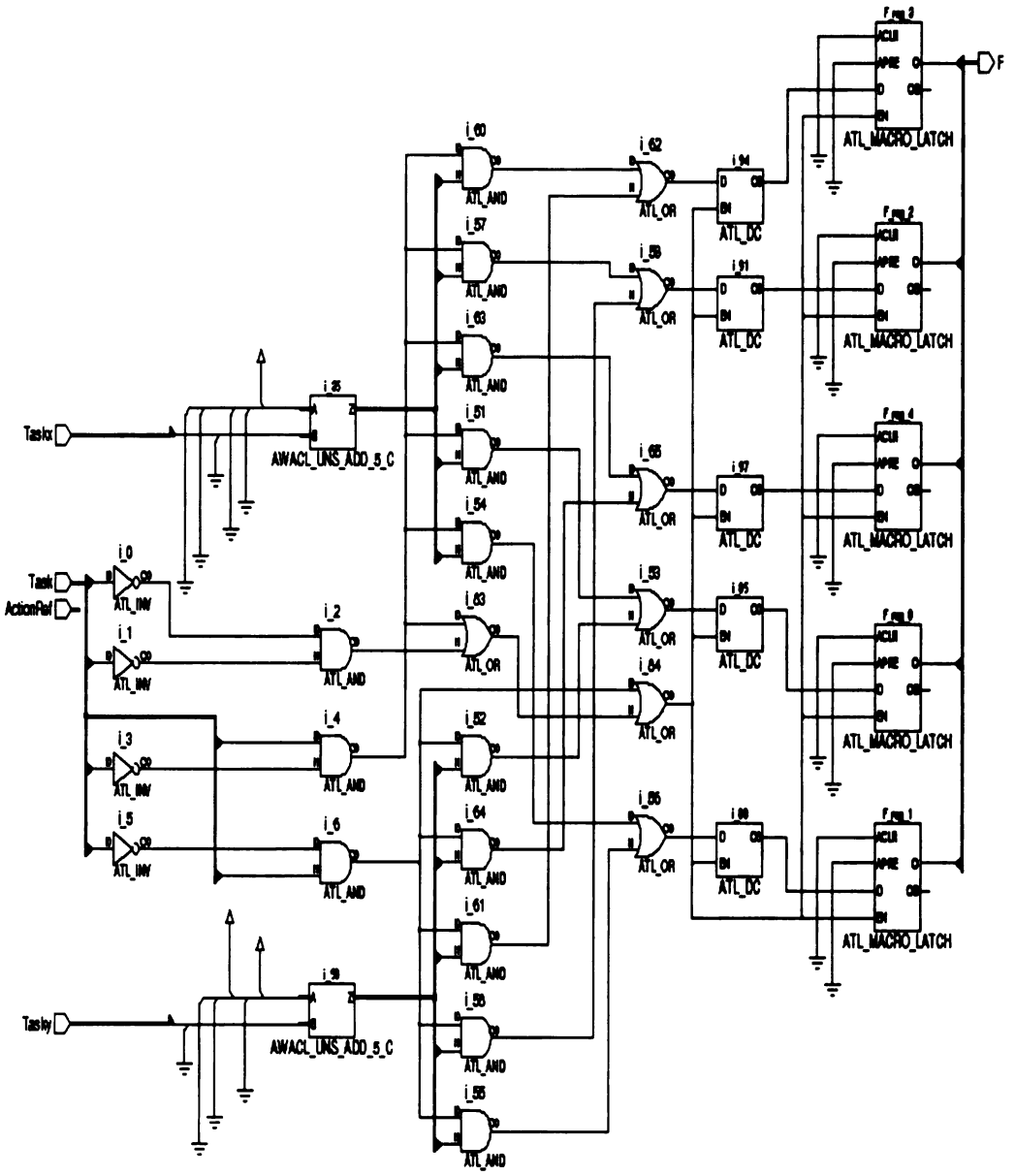


Figure 7.8: Schematic Description of Action Planner.

the simulation, it shows the changes of the task reference, the task input, the action reference and the action input and the motion reference. The system performs only one task. The task is to move the robot to destination point (3,4). From Figure 7.10, we can see that the perceptive references triggered the system at point(3,0), when the robot traveled along X direction for 3, i.e. at the moment 3s, the wall signal becomes 1, the action reference and command switch from 1 to 2, the second action command

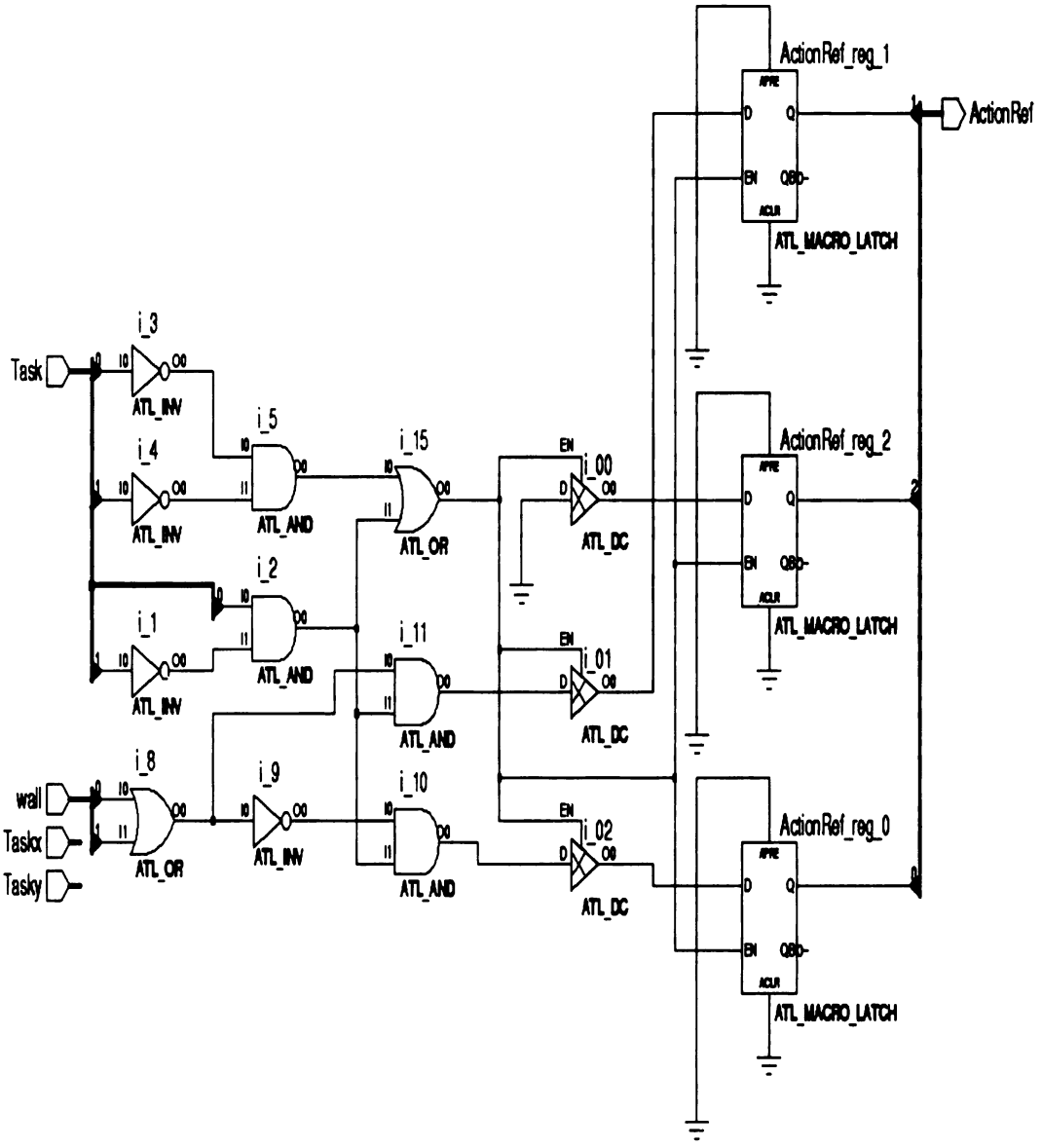


Figure 7.9: Schematic Description of Action Reference.

is issued, which means the motion on Y direction, along the wall, the action command and its parameters will be held for the entire period of the action execution, until the task has been executed.

The second task is to show the hybrid properties of the perceptive controller, we use the motion planner to compute the trajectory through the arithmetic operations. Instead of two segments of straight line, the first segment is a circular path which is

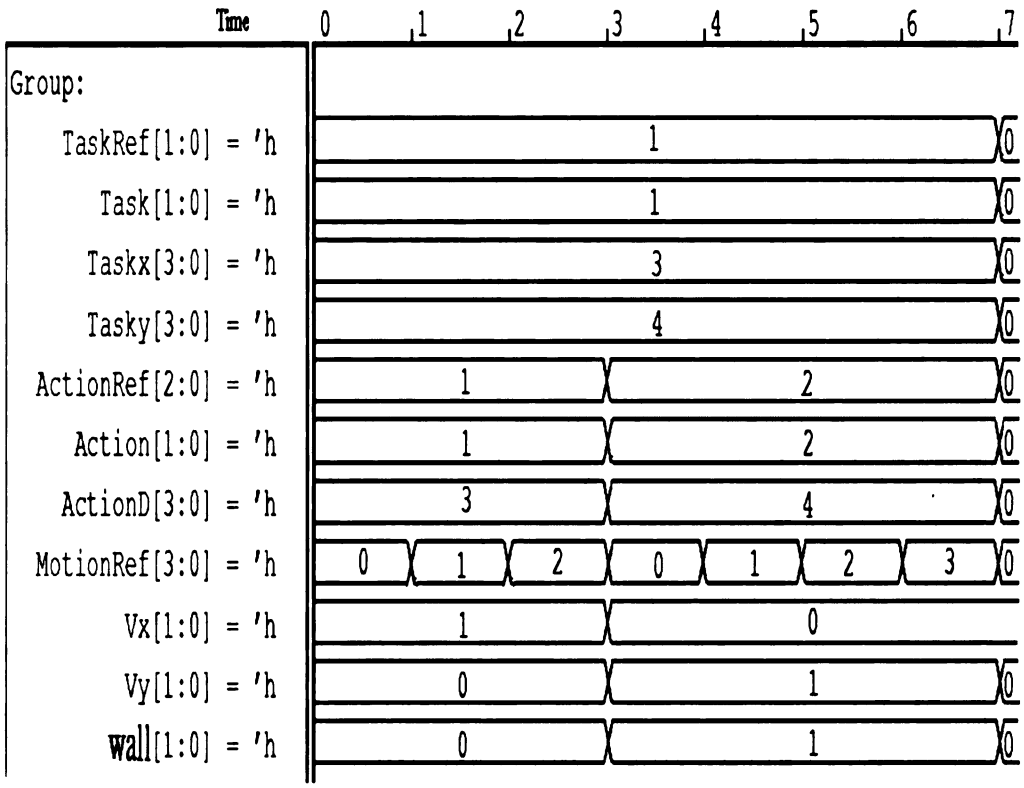


Figure 7.10: Simulation results.

implemented by the designed arithmetic unit. In perceptive frame the path planning algorithm can be described as a circular path:

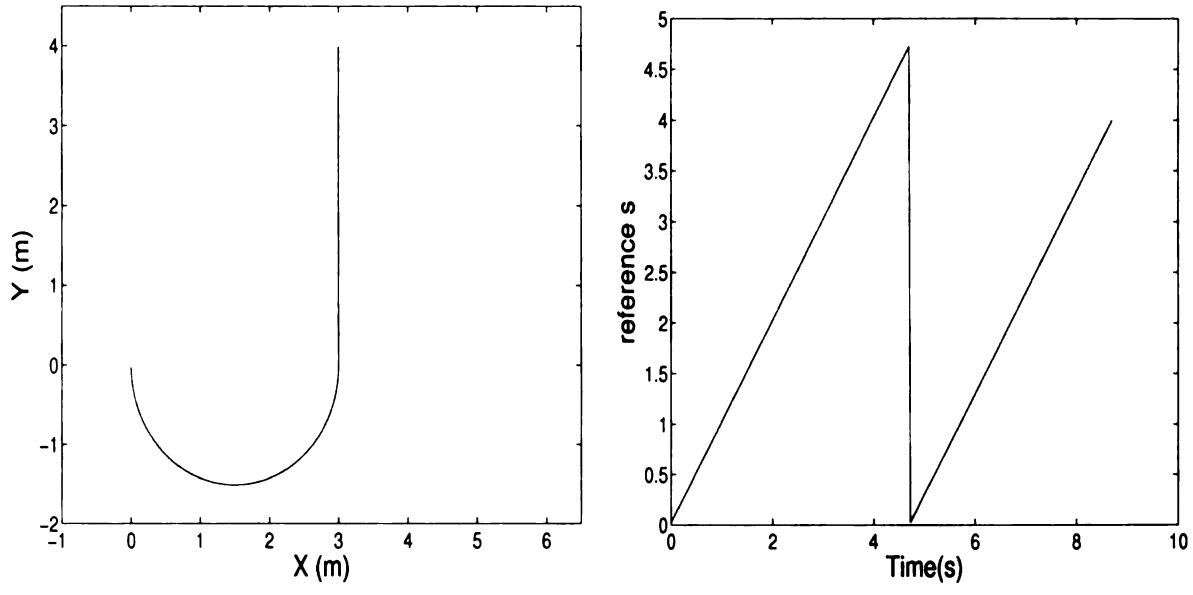
$$x = rcos(s/r) \tag{7.11}$$

$$y = rsin(s/r) \tag{7.12}$$

$$\dot{x} = -\frac{y}{r}V(s) \tag{7.13}$$

$$\dot{y} = \frac{x}{r}V(s) \tag{7.14}$$

Where r is the radius of the circular path, s is the continuous motion reference. Figure 7.11 shows the trajectory in the simulation, and the continuous motion reference w.r.t. time, which is computed by the continuous dynamics model and the motion reference block.



(a) (b)

Figure 7.11: The Motion Trajectory in the Experiment: (a) x-y plot (b)continuous reference w.r.t time.

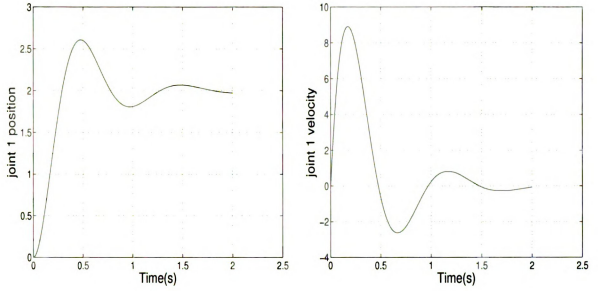
For a robot arm, dynamics model is given by

$$D(q)\ddot{q} + C(q,\dot{q}) + G(q) + J_h^T(q)f_e = \tau \tag{7.15}$$

Using the above model as the continuous model of the robot, one can build a second order control hybrid simulation. The D, H, C can be calculated in hardware part for generating the computed torques τ. The following is the hybrid simulation results based the second order robotic control simulation model. The figures indicate the step response for the individual joints. Puma 560 first 3-joint model is used for the simulation in Test Builder. The control signal is to move the joint for 2 degrees.

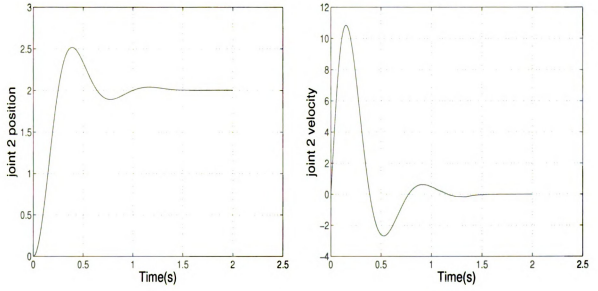
Figures 7.12, 7.13, and 7.14 show the step response of joint level control as a second order control system. It can be seen that joint 1, 2, 3 can track the step signal accurately. The VLSI design can satisfies the real time requirements under the given PD control gains.

Figures 7.15, and 7.16, and 7.17 show the joint level control of joint 1, 2, and 3



(a) (b)

Figure 7.12: Joint Level Control – Step Response: (a) position (b) velocity

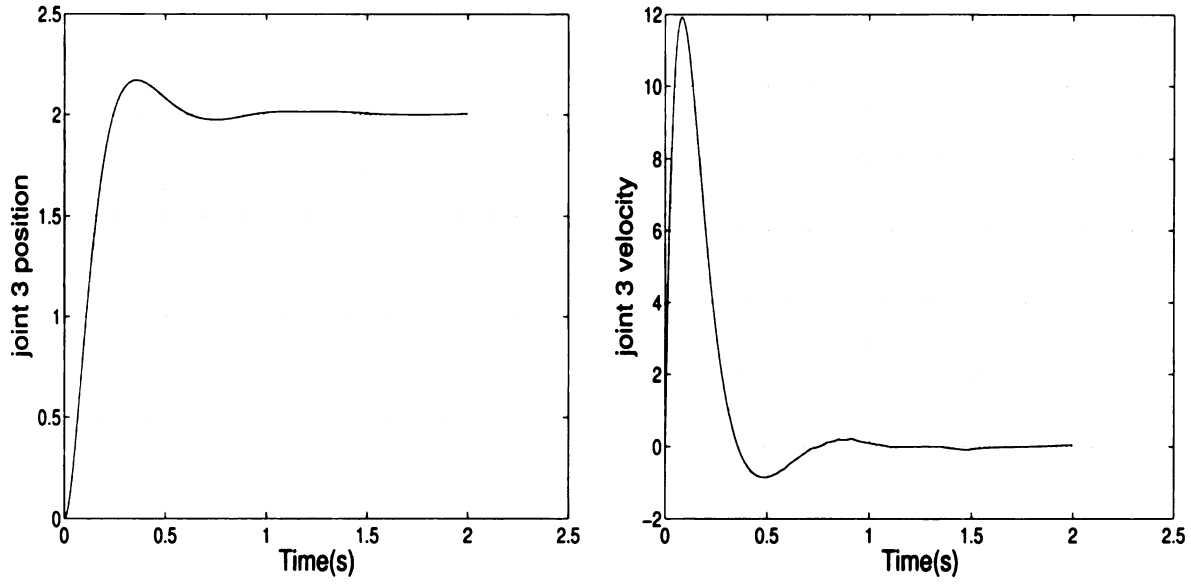


(c) (d)

Figure 7.13: Joint Level Control – Step Response: (c) position (d) velocity

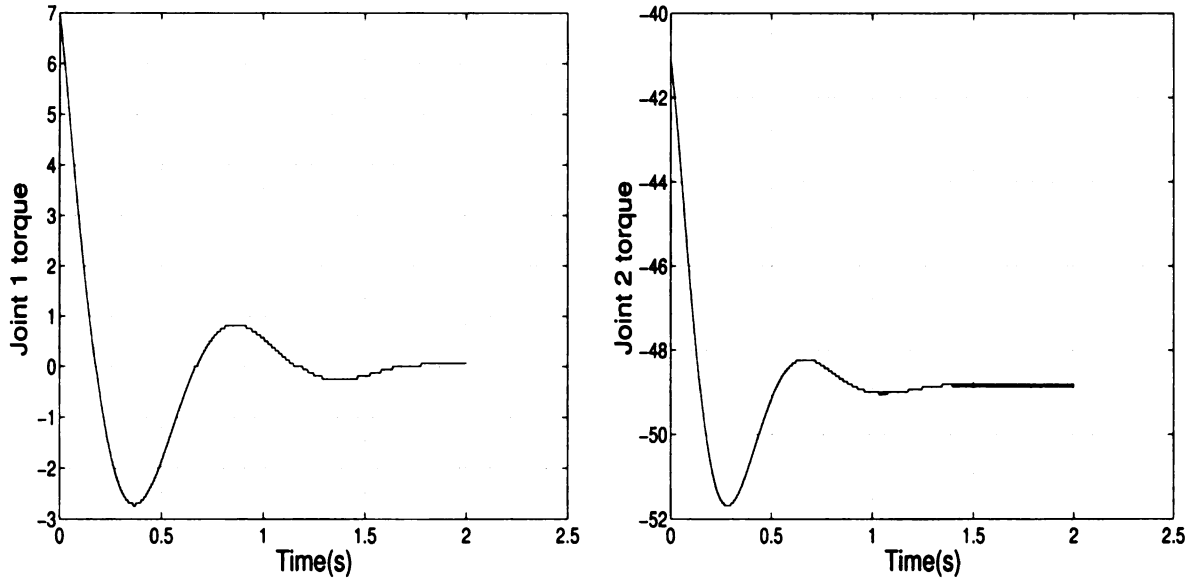
with 14 bits long decimal part. The figures indicate that the system does not have significant improvement in stability and tracking accuracy.

Figures 7.18, and 7.19, 7.20 and 7.21 indicate the step response of the joint level control of joint 2 with 8 bits long decimal part. The figures show that the system still



(e) (f)

Figure 7.14: Joint Level Control – Step Response: (e) position (f) velocity



(g) (h)

Figure 7.15: Joint Level Force Control Signal at Step Response: (g) Joint 1 (h) Joint 2

compare to the results with 24 bits long arithmetic units, the accuracy of the tracking control is significantly decreased because of the limited length of the arithmetic operation. In this case, the control force has jumps and discountinuities, and the velocity on this joint is not smooth. Thus, it can be seen that 24-bit arithmetic unit

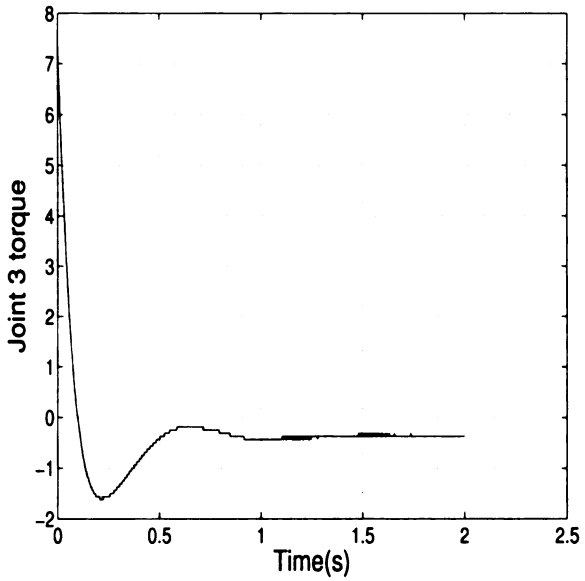
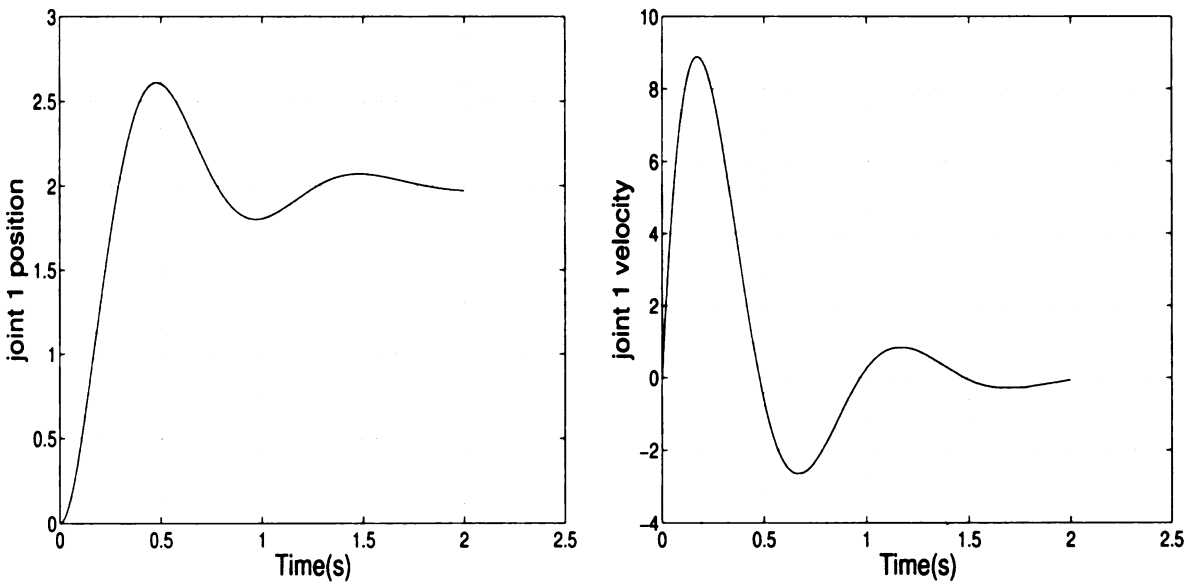


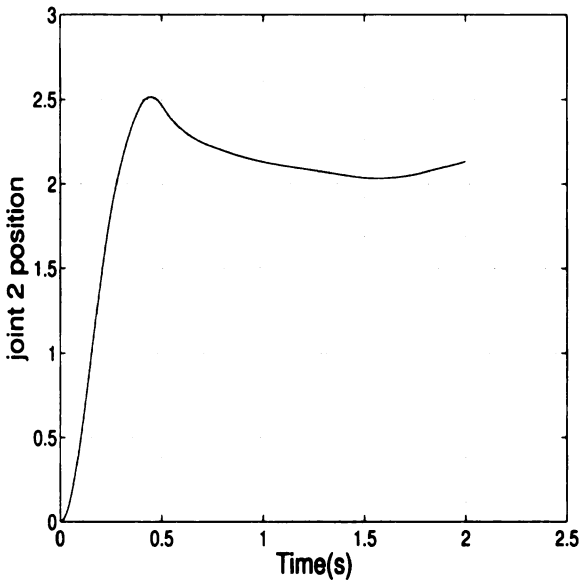
Figure 7.16: Joint Level Force Control Signal at Step Response: Joint 3 .



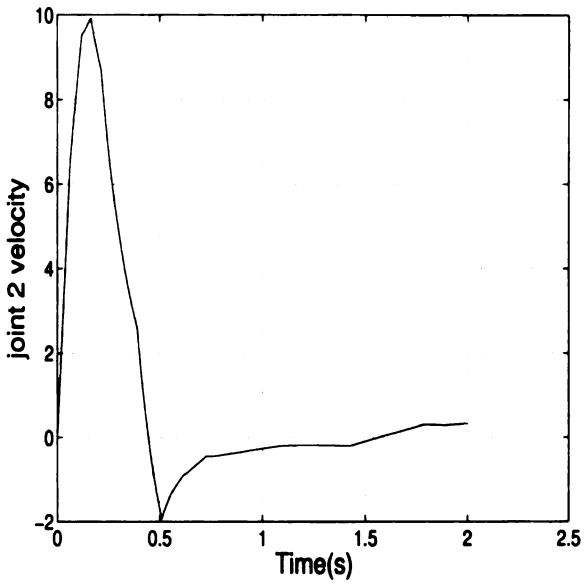
(a) (b)

Figure 7.17: Joint Level Control Step Response: (a) position (b) velocity

can provide good performance in tracking accuracy, and the time requirement can be satisfied.

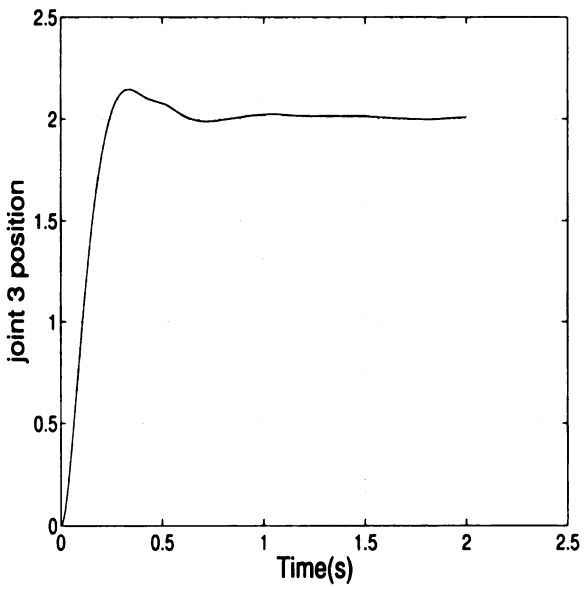


(c)

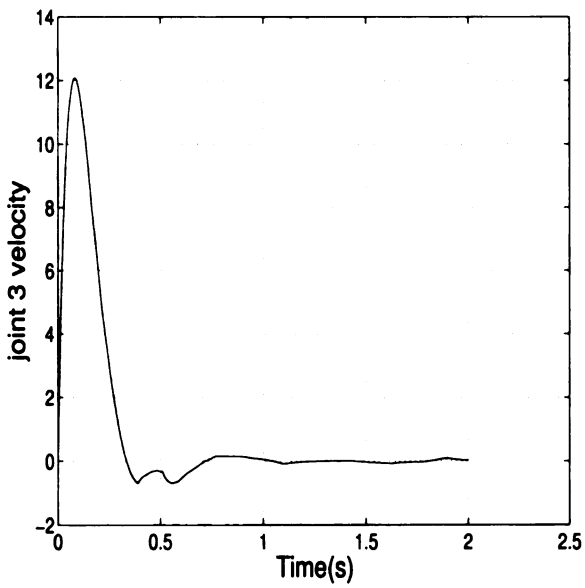


(d)

Figure 7.18: Joint Level Control Step Response: (c) position (d) velocity

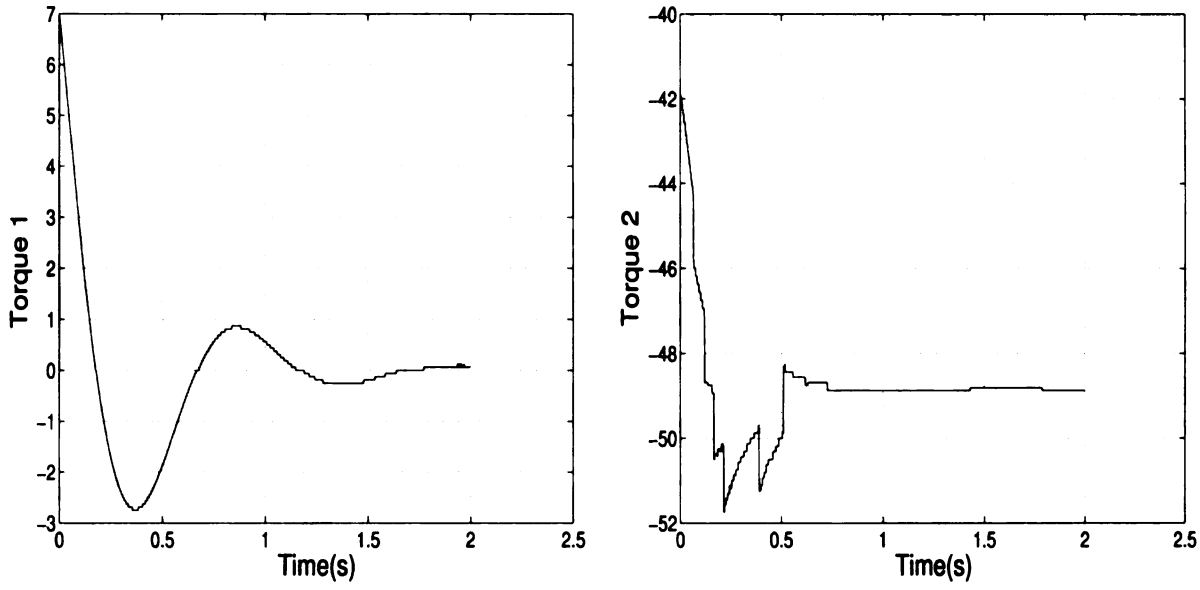


(e)



(f)

Figure 7.19: Joint Level Force Control Step Response: (e) position (f) velocity



(g) (h)

Figure 7.20: Joint Level Control at Step Response: (g) Position of Joint 2 (h) Velocity of Joint 2

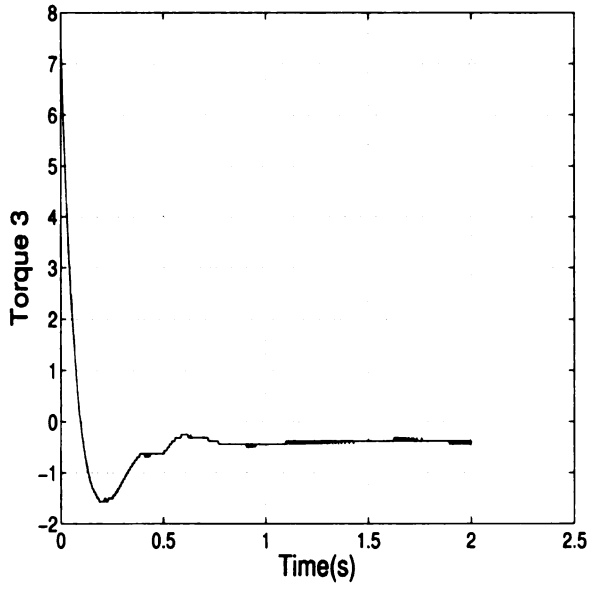


Figure 7.21: Joint Level Control Signal at Step Response (i) : Joint 2 .

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

This dissertation describes the Design and VLSI implementation of Perceptive Controller for Robotic Systems. The research covers task model learning, hybrid control system modeling, analyzing, and designing. The proposed method is an efficient approach to explore hybrid systems. The main contributions of this dissertation are as follows:

Task Modeling in Perceptive Frame:

A Wavelet Based Model Identification Method has been introduced. The experimental results show the effectiveness of the proposed method for identifying the mass and the length of a nonholonomic cart by interactive action in cart pushing, when the unknown noises generated by sensor measurements and numerical operations are uncorrelated.

The significance of the task modeling method is that the method offers significant advantages over the classical least square estimation methods. In model identification for online estimation without prior statistical knowledge of measurement and operation noises. Many applications, including home care, search, rescue, require the mobile manipulator working in unstructured environments. Based on the method proposed in this dissertation, the task parameters can be estimated by simple interactive actions between the mobile manipulator and the environment. It improves the effectiveness of the operation significantly.

Linguistic Control Approach:

A hybrid model for the perceptive reference based robotic plan and control systems has been proposed. The model integrates the continuous perceptive reference and discrete references through a hybrid approach and offers significant advantages over

classical time based schedule-plan-control models. Based on the properties of the proposed model, some properties of the system, including evolution of the perceptive reference, and stability are discussed.

The hybrid model makes the system become more intelligent and flexible in the processing of environmental sensory measurements for planning and control. The experimental results show the effectiveness of the proposed model for processing unexpected event, These results are significant in the modification of the designed plan and the mechanism which triggers multiple discrete events. The intelligence of the system has been enhanced.

Perceptive Control System Design:

In order to implement real-time systems in the perceptive frame, an implementation-oriented design method has been introduced for robot controllers. Considering the hybrid system model in perceptive frame, the functional abstraction of the model has been analyzed. The high level architecture of the model makes the controller modularized, therefore, synthesizability can be guaranteed. An implementation oriented controller design method is proposed for describing the system model. The automaton based description of the robotic control system can be performed by the code in Verilog, which is semantically equivalent to the controller model. The significant impact is that the implementation-oriented design language provides a convenient way for designing a real-time system. The language is used to convert the control system design to hardware implementation via semantics description of the tasks. The perceptive frame control can be applied to other applications in real time control area such as motion control systems. The Verilog code has been tested by Candence design system to be synthesized to a structural description and simulated in a hybrid fashion. The simulation results show that based on the hybrid control model in perceptive frame, the implementation oriented design method can translated the high level description to low level VLSI implementation.

8.2 Suggestions for Future Research

The research work in the dissertation has only opened the door little wider to future research investigation. In the future, there will be some areas of the analysis and design where future research would be very beneficial. Some suggestions are as follows:

- **Model Learning**: The Wavelet based online model learning method can be extended to multiple parameter identification for more complex model learning. Theoretical analysis will be an important argument to show the estimation convergence in subbanding the raw signal. Also, Kalman filter can be applied to state estimation with nonholonomic constraints.

- **System Modeling and Analysis**: This dissertation presents a framework for investigation of hybrid system in perceptive frame. In this frame work, some theoretical analysis and design problems will be very beneficial. Context Free-type and recursive enumerable grammars can be used in the liguistic expressions at task level and action level, to gain more flexibility and capacity of tast-action description. The unexpected events, perceptive reference and commands can compose the homogenuous linguistic expression, some properties, including non-uniformly sampling, complexity and dynamics can be extensively discussed.

- **Real Time System Implementation on a Chip (ASIC)**: The Cadence / testbuilder design framework provides an approach for simulating a chip-based hybrid control system. The future research will be on the silicon implementation of the real time system. Hence, optimization of algorithm, the sice of arithmetic units and testing design will be the major topics.

- **Software/Hardware Co-Design**: Based one the hybrid system model in perceptive frame, the design and implementation procedures have been unified by the framework of automata and linguistic descriptions. The software/hardware

co-design technique can be developed based on the hybrid perceptive framework. Hardware and software compilation at difference level will be discussed for task/action assignment in the sense of real time control with the final goal of optimizing partition plan obtained by analyzing the resource and performance of the system.

BIBLIOGRAPHY

[1] Mark J.T.Smith "Subband and Wavelet Transforms: Design and Applications"Kluwer Academic Publishers, 1996.

[2] R. Alur, C. Courcoubetis, and T.A.Henzinger, "Hybrid Automata: An Algorithmic Approach to the specification and Verification of Hybrid Systems", *Proceedings of the IEEE* Vol.88, NO.7, July 2000.

[3] Panos J. Antsaklis. M. D. Lemmon. "Hybrid Systems Modeling and Autonomous Control Systems"volume 736 of *Lecture Notes in Computer Science,*Pages 366-392, Springer-Verlag, New York, 1993.

[4] Allen Back, John Guckenheimer, and Mark Myers. "A Dynamical simulation facility for hybrid systems"volume 736 of *Lecture Notes in Computer Science,* pages 255-267, Springer-Verlag, New York, 1993.

[5] Michael S. Braniky, "Studies in Hybrid Systems: Modeling, Analysis, and Control", Ph.D Thesis of M.I.T. 1995.

[6] Michael S. Braniky, "Multiple Lyapunov Functions and Other Analysis Tools for Switched and Hybrid Systems", *IEEE Transactions on Automatic Control,* Vol 43, No.4,1998,pp.475-482.

[7] Michael S. Braniky, Vivek S. Borkar, and Sanjoy K. Mitter, "A Unified Framework for Hybrid control", *Proceedings of the 33rd Conference on Decision and Control,* pp.4228-4234, Lake Buena Vista, FL-December, 1994.

[8] R. Brockett, "On the Computer Control of Movement" *Proceedings of the 1988 IEEE Conference on Robotics and Automation,* New York, April 1988, pp. 534-540.

[9] R. Brockett, "Language Driven Hybrid Systems" *Proc. of the 33rd Conference on Decision and Control,*Lake Buena Vista,FL,Dec.1994.

[10] R. W. Brockett. "Hybrid Model for Motion Control Systems." in *Perspectives in Control.* Editors. H. Trantelman and J.C.Willems, pp. 29-54, Birkhauser, Boston, 1993.

[11] R. Brockett, "Formal Languages for Motion Description and Map Making," in *Robotics*, (R.W. Brockett, ed.) American Mathematical Society, Providence, RI, 1990. pp. 181-193.

[12] Alan Burn and Andy Wllings, "Real-Time Systems and Programming Languages (Third Edition) Ada 95, Real-Time Java and Real-Time POSIX", Addison Wesley Longmain, March, 2001.

[13] John F. Canny, "The Complexity of Robot Motion Planning". The MIT Press, 1987.

[14] Andrew Choi, "Real-Time Fundamental Frequency Estimation by Least-Square Fitting", *IEEE Trans. On Speech and Audio Processing,* Vol. 5, No. 2, pp201-pp205, March 1997.

[15] Ingrid Daubechies, "Ten lectures on wavelets". Society for Industrial and applied mathematics. Philadelphia, 1992. *Notes from the 1990 CBMS-NSF Conf.* Wavelets Applications, Lowell, MA, USA.

[16] M. Egerstedt, "Linguistic Control of Mobile Robots" *IEEE/RSJ IROS* Maui, Hawaii, USA, Oct. 2001.

[17] M. Egerstedt "Behavior Based Robotics Using Hybrid Automata" *Lecture Notes in Computer Science: Hybrid Systems III* pp. 103-116, Springer-Verlag, March 2000.

[18] M. Egerstedt and Roger W. Brockett "Feedback Can Reduce the specification Complexity of Motor Programs" *IEEE TRANSACTIONS ON AUTOMATIC CONTROL* pp 213-223, VOL.48.NO.2, FEBRUARY 2003.

[19] Imad Elhajj, Jindong Tan, Yu Sun and Ning Xi, Supermedia Enhanced Human/Machine Cooperative Control of Robot Formations, IEEE/RSJ International Conference on Intelligent Robots and Systems, October, 2002, EPFL, Switzerland

[20] Vashti Christina Galpin "Equilablence semantics for concurrency: comparison and application". Ph. D. Thesis, University of Edinburgh, 1998.

[21] K. S. Fu, R. C. Gonzalez, and E. S. G. Lee "ROBOTICS - Control, Sensing, Vision, and Intelligence". McGRAW-HILL INTERNATIONAL EDITIONS, Industrial Engineering series, USA.

[22] Arthur A. Giordano, Frank M.Hsu, "Least Square Estimation with Application to Digital Signal Processing", A Wiley-Interscience Publication 1985.

[23] Mohinder S. Grewal and Angus P. Andrews, "Kalman Filtering, theory and practice", Prentice Hall Information and System Sciences series, Thomas Kailath, Series Editor Englewood Cliffs, new Jersey, 1993.

[24] R. L. Grossman, Anil Nerode, A. P. Ravn, and H. Rischel. editors. "Hybrid Systems", volume 736 of *Lecture Notes in Computer Science,* Springer-Verlag, New York, 1993.

[25] Thomas A. Henzinger. "The theory of hybrid automata", *Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press*, 1996, pp. 278-292.

[26] Thomas A. Henzinger. "The theory of hybrid automata" *Verification of Digital and Hybrid Systems (M.K. Inan, R.P. Kurshan, eds.), NATO ASI Series F: Computer and Systems Sciences*, Vol. 170, Springer-Verlag, 2000, pp. 265-292.

[27] Felix Hausdorff. *Set Theory.* Translated from the German by John R. Aumann, ET AL, Chelsea Publishing Company, New York, N.Y, 1957.

[28] J.E.Hopcroft. R. Motwani and J. D. Ullman, "Introduction to Automata Theory, Languages, and Computation." 2nd Edition, Addison Wesley, 2001,

[29] Ling Hou, Anthony. N Michel and Hui Ye, "Stability Analysis of Switched Systems" *IEEE Conference on Decision and Control,* December 1996.

[30] T. C. Hsia, "System Identification: Least Square Method". Lexington Books, 1974.

[31] R. Isermann "practical aspects of process identification",*Automatica* ,1982, Vol. 16. pp.575-587.

[32] M. Kam, X. Zhu and P. Kalata, "Sensor Fusion for Mobile Robot Navigation", *Proceedings of the IEEE*, pages 108-119, vol. 85, No. 1, 1997.

[33] Hassan K. Khalil "Nonlinear Systems", second edition, Prentice Hall, Upper Saddle River, New Jersey,1996. ‘

[34] Akram Ladroubi, and Karlheinz Grochenig "Nonuniform Sampling and Reconstruction in Shift-invariant Spaces" *SIAM Review,* Vol. 43, No.4, pp. 585-620.

[35] W. Li and J.-J.E.Slotine , "Parameter Estimation Strategies for Robotic Applications" *A.S.M.E Winter Annual Meeting* 1987.

[36] J. Lygeros, K. H. Johansson,S. N. Simic, J. Zhang, and S. S. Sastry "Dynamical Properties of Hybrid System", *IEEE Transactions On Automatic Control* Vol. 48, No. 1, pp2-pp17, JANUARY 2003.

[37] J. Lygeros, Karl Henrik Johansson, Slobodan N. Simic, Jun Zhang, Shankar Sastry. "Continuity and Invariance of Hybrid Automata" *Proceedings of the 40th IEEE Conference on Decision and Control,* Page 340-345, Orlando, Florida, December 2001.

[38] J. Lygeros, Karl Henrik Johansson, Slobodan N. Simic, Jun Zhang, Shankar Sastry. On the Existence of Executions of Hybrid Automata *Proceedings 38th IEEE Conference on Decision and Control(CDC'99)*, Phoenix, Arizona, December 1999.

[39] V. Manikonda, P.S. Krishnaprasad and J. Hendler, "Behaviors, Hybrid Architectures and Motion Control." *In Mathematic Control Theory.* pp.199-226,Springer-verlag,1998,

[40] Anthony N. Michel, "Recent Trends in the Stability Analysis of Hybrid Dynamical Systems" *IEEE* VOL. 45, NO.1, JANUARY 1999.

[41] Alan V. Oppenheim, Ronald W. Schafer "Discrete-Time Signal Processing", Prentice hall, Englewood Cliffs, New Jersey, 1989.

[42] Robi Polikar "The engineer's ultimate guide to wavelet analysis, the wavelet tutorial", http://engineering.rowan.edu/ polikar/WAVELETS/WTtutorial.html

[43] Stuart J. Russell and Peter Norvig, "Artificial Intelligence", Prentice-Hall, 1995.

[44] Hisashi Sasaki, "A Formal Semantics for Verilog-VHDL Simulation Interoperability by Abstract State Machine" *Design, Automation and Test in Europe (DATE)*, March, 1999.

[45] Robert W. Sebesta, "Concepts of Programming Languages", Addison-Wesley Publishing, 5th edition, July, 2001.

[46] R. Sermann and U. Baur, "Two Step Process Identification with Correlation Analysis and Least Squares Parameter Estimation", *Trans. of ASME, Series G.J. of Dynamic Systems measurement and Control* Vol.96,pp.425–432, 1974.

[47] Zoran Salcic "VHDL AND FPLDs: IN DIGITAL SYTEMS DESIGN, PROTOTYPING AND CUSTOMIZATION", KLUWER ACADEMIC PUBLISHERS. 1998.

[48] Calude E. Shannon, "Classic paper: Communication in the Presence of Noice", *Proceedings of the IEEE,* VOL. 86, NO.2, FEBRUARY, 1998.

[49] M. Song, Tzyh-Jong Tarn and N. Xi, "Integration of Task Scheduling, Sensing, Planning an Control in Robotic Manufacturing system", *Proceedings of the IEEE* Vol.88, NO.7, July 2000.

[50] Mumin Song, Tzyh-Jong Tarn, and Ning Xi "Integrated Hybrid System Approach for Planning and Control of Concurrent Tasks in Manufacturing Systems". *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1192-1197,volume 2, Leuven, Belgium, May 16-20, 1998.

[51] Yu Sun, Ning Xi, Jindong Tan, and Yuechao Wang Interactive Model Identification for Nonholonomic Cart Pushing by a Mobile Manipulator , proceedings of IEEE ICRA 2002, Washington D.C.

[52] Yu Sun, Ning Xi, Jindong Tan, "On-line Model Learning for Robotic Manipulations"IEEE International Conference on Mechatronics and Machine Vision in Practice, Thailand.

[53] Yu Sun, Ning Xi, and Jindong Tan, A Hybrid System Approach for Event-Based Planning and Control of Robotic Operations, accepted by "Computational Intelligence in Robotics and Automation" (IEEE CIRA) 2003.

[54] Jindong Tan. and Ning Xi, "Unified Model Approach for Planning and Control of Mobile Manipulators", *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3145-3152, Seoul, Korea, May, 2001.

[55] Jindong Tan "Control of Mobile Manipulators", Ph.D Thesis, Michigan State University in East Lansing, 2002.

[56] Tzyh-Jong Tarn, Mumin Song and Ning Xi "Intelligent planning and control for hybrid system" *Intelligent Robots and Systems, 1998. Proceedings, 1998 IEEE/RSJ International Conference on* , Volume: 2 , 13-17 Oct 1998 Page(s): 972 -977 vol.2

[57] Lucio Tavernini. "Differential automata and their discrete simulators",*Nonlinear Analysis, Theory, Methods, and Applications* 11(16):665-683, 1987.

[58] Michael Unser, "Sampling-50 year after Shannon", *PROCEEDINGS OF THE IEEE,* Vol.88,NO.4, APRIL, 2000.

[59] Hans S. Witsenhausen. "A class of hybrid-state continuous-time dynamic systems" *IEEE Transactions on Automatic Control*,11(2):161-167, 1966

[60] N. Xi, "Event-Based Motion Planning and Control for Robotic Systems" . Ph.D thesis, Washington Univ. in St. Louis,1993.

[61] Ning Xi, Tzyh-Jong Tarn and A. K. Bejczy "Intelligent Planning and Control for Multirobot Coordination: An Event-Based Approach", *IEEE Trans. on Robotics and Automation* Vol.12, NO.2,June 1996.

[62] Ning Xi, Tzyh-Jong Tarn, and A. K. Bejczy, "Intelligent Planning and Control for Multirobot Coordination: An Event-Based Approach", *IEEE Trans. on Robotics and Automation*, pages 439-452, vol. 12, No. 3, 1996.

[63] Ning Xi, Tzyh-Jong Tarn, "Stability Analysis of non-time referenced internet-based telerobotic systems", *Robotics and Autonomous Systems*, pages 173-178, vol. 32, 1996.

[64] Ning Xi and Tzyh-Jong Tarn "Modeling and control of multiple segment robotic operations in a perceptive reference frame" *Advanced Intelligent Mechatronics '97., IEEE/ASME International Conference on* , 16-20 Jun 1997 Page(s): 76

[65] Ning Xi and Tzyh-Jong Tarn "Integrated task scheduling and action planning/control for robotic systems based on a max-plus algebra model" *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on* , Volume: 2 , 7-11 Sep 1997 Page(s): 926 -931 vol.2

[66] Y. Yamamoto "Control and Coordination of Locomotion and Manipulation of a Wheeled Mobile Manipulators, Ph.D Dissertation in University of Pennsylvania. August 1994.

[67] Hui Ye, Anthony N. Michel and Ling Hou "Stability Theory for Hybrid Dynamical System", *IEEE Transactions On Automatic Control* Vol. 43, No. 43, pp461-pp474, APRIL 1998.

[68] H. Zhuang, Zvi S. Roth "A Linear Solution to the Kinematic Parameter Identification of Robot Manipulators" *IEEE Transactions on Robotics and Automation* Vol. 9, No. 2, April 1993.

[69] Stephen Edwards, Luciano Lavagno, Edward A. Lee, and Alberto Sangiovanni-Vincentelli, "Design of embedded Systems: Formal Models, Validation, and Synthesis". *Proceedings of the IEEE,* VOL.85, No. 3, March, 1997,pp366-390.

[70] Giovanni De Micheli, "Synthesis and Optimization of Digital Circuits" *McGraw-Hill, Inc.* 1992.

[71] Petra Michel Ulrich Lauther Peter Duzy, "The Synthesis Approach to digital System Design", Kluwer Academic Publishers, 1992.

[72] Zoran Salcic, "VHDL and FPLDs In Digital Systems Design, Prototyping and Customization", *Kluwer Academic Publishers* 1998.

[73] Mark Gordon Arnord "Verilog Digital Computer Design: Algorithms into Hardware", Upper Saddle River, 1999.

[74] Nelio Muniz Mendes Alves and Sergio de Mello Schneider "Implementation of an embedded Hardware Description Language Using Haskell". *Journal of Universal Computer Science,* Vol. 9, No. 8(2003), 795-812.

[75] Catherine G. Wong and Alain J. Martin "Data-driven process decomposition for circuit synthesis". *Proceedings of IEEE Conference on Electronic Circuits and Systems*, September 2001.

[76] Reid Harrison, Christof Koch, "Guiding a robot with an Analog VLSI motion sensor Based on the Visual System of the Fly", Technical Report at Center for Neuromorphic Systes Engineering, Caltech, 2002.

[77] Jorgen Staunstrup and Wayne Wolf, "Hardware/Software Co-Design": Priciples and Practice, KLUWER ACADEMIC PUBLISHERS, BOSTON/DORDRECHT/LONDON, 1997.

[78] Toshio Takahashi, "New Digital Hardware Control Method for High Performance AC servo Motor Drive-Accelerator Servo Drive Development Platform for Military Application", *Military Electronics Conference*, Sept. 24-25, 2004.

[79] Fabrice Aubepart, Philippe Poure and Francis Braun, "VLSI Design Approach of Complex Motor Control, Case of Direct Torque Control of AC machine" . *The 7th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2000., Volume: 2, 2000* pp. 814 817.

[80] Daniel D. Gajski, Frank Vahid, Sanjiv Narayan and Jie Gong, "Specification and Design of Embedded System", PTR Prentice Hall, Englewood Cliffs, New Jersey, 1994.