Study on Automotive Embedded System Design of Engine, Brake and Security System

自動車のエンジン・ブレーキ・セキュリティ
組み込みシステム設計に関する研究

2006年4月

早稲田大学大学院情報生産システム研究科
情報生産システム工学専攻
プロセス制御研究

解 龍
## CONTENTS

**Chapter 1 Introduction** ..................................................................................1

1.1 Background of Automotive Embedded Systems ......................................... 1

1.2 Motivation and Objective of the Research ................................................... 2

1.3 The Organization of This Dissertation ....................................................... 3

**Chapter 2 Points of Current Research** .....................................................5

2.1 Embedded System Design for Engine Control System ................................. 5

2.2 Embedded System Design for ABS and Development Tools ....................... 7

2.3 Embedded System Design for Automotive Security ..................................... 10

**Chapter 3 Hybrid Embedded Design of Engine Intake System ...............14**

3.1 Mathematical Model of Hybrid System ....................................................... 14

3.2 Preliminary of Hybrid Simulation Algorithm ............................................... 19

3.2.1 Concepts and Structures of Hybrid Simulation ....................................... 19

3.2.2 Origins and Definitions of the Algorithm ............................................. 21

3.2.3 Numerical Integration for Continuous Locations .................................. 24

3.3 Hybrid Simulation Algorithm Design ......................................................... 26

3.3.1 Mechanism and Representation of Event .............................................. 26

3.3.2 Event Detection and Location ............................................................... 29

3.3.3 Integration Formula on the Sliding Surface .......................................... 41

3.3.4 Hybrid Automaton Expansion and Flow ............................................... 44

3.4 Modeling and Simulation of Intake System ................................................. 49

3.4.1 Basics of Turbocharged Internal Combustion Engine ............................ 49

3.4.2 Intake System Model of Turbocharged Engine ...................................... 52

3.4.3 Simulation of Intake System Model ....................................................... 56

3.4.4 Intake System Implementation .............................................................. 68

3.5 Summary .................................................................................................... 72

**Chapter 4 Anti-lock Brake System Design** ..............................................73

4.1 Vehicle Model and Simulation on Software Platform .................................. 73
Chapter 1 Introduction

1.1 Background of Automotive Embedded Systems

Every year, automobile manufacturers worldwide pack new embedded system into their vehicles. Tiny processors under the hood and in the deep recesses of the car gather and exchange information to control, optimize, and monitor many of the functions that just a few years ago were purely mechanical. The technological advancements of embedded system and electronics within the vehicle are being driven by the challenge to make the vehicle safer, more energy efficient and networked. Flash-based microcontrollers, from on-chip system to FPGA, are the command center for embedded system design.

In 1968, the Volkswagen 1600 used a microprocessor in its fuel injection system, launching the first embedded system in the automotive industry. Historically, low-cost 8- and 16-bit processors were the norm in automotive controllers, and software engineers developed most of the code in assembly language. However, today's shorter development schedules and increased software complexity have forced designers to resort to select the more advanced CPUs and a higher level language in which designers can easily reuse modules from project to project. A successful automotive-electronic design depends on careful processor selection. Modern power train controllers for the engine and transmission generally require 32-bit CPUs to process the real-time algorithms. Other areas of the automobile, such as safety, chassis, and body systems, use both 16-bit and 32-bit processors, depending on control complexity. Although some critical timing situations still use assembly language, the software trend in automotive embedded systems is toward C. The control software is more complicated and precise for the current vehicles.

Advanced usage of embedded system and electronics within the vehicle can aid in controlling the amount of pollution being generated and increasing the ability to provide systems' monitoring and diagnostic capabilities without sacrificing safety/security features that consumers demand. The electronic content within the vehicle continues to grow and more systems become intelligent through the addition of microcontroller based electronics. A typical vehicle today contains an average of 25-35 microcontrollers with some luxury vehicles containing up to 70 microcontrollers per vehicle. Flash-based microcontrollers
are continuing to replace relays, switches, and traditional mechanical functions with higher-reliability components while eliminating the cost and weight of copper wire.

Embedded controllers also drive motors to operate power seats, windows, and mirrors. Driver-information processors display or announce navigation and traffic information along with vehicle diagnostics. Embedded controllers are even keeping track of your driving habits. In addition, enormous activity occurs in the entertainment and mobile-computing areas. Networks are a recent addition to embedded controllers which are the challenge of squeezing in the hardware and code for in-car networking. To satisfy new government emissions regulations, vehicle manufacturers and the Society of Automotive Engineers (SAE) developed J1850, a specialized automotive-network protocol. Although J1850 is now standard on US automobiles, European manufacturers support the controller-area network (CAN). High-bandwidth, real-time control applications like power train, airbags, and braking need the 1Mbps speed of CAN and their safety critical nature requires the associated cost. Local Interconnect Network (LIN) typically is a sub-bus network that is localized within the vehicle and has a substantially lower implementation cost when compared to a CAN network. It serves low-speed, low-bandwidth applications like mirror controls, seat controls, fan controls, environmental controls, and position sensors.

Embedded system in the automotive shares the general characters of common embedded system, but it has its own primary design goals of automotive industry. Reliability and cost may be the toughest design goal to achieve because of the rugged environment of the automobile. The circuitry must survive nearby high-voltage electronic magnetic interference (EMI), temperature extremes from the weather and the heat of the engine, and severe shock from bad roads and occasional collisions. The electronic control units (ECUs) should be developed and tested on the all kinds of situations with low cost. Although testing time grows with the complexity of the system, a reliable controller also requires complete software testing to verify every state and path. A single bug that slips through testing may force a very expensive recall to update the software. Therefore the development of high-ability tools is also active in the field of automotive embedded system.

### 1.2 Motivation and Objective of the Research

Being the core of automotive electronic and control system, the combination of ECUs continues to advance tomorrow’s automobiles with the ability to provide the driver with a safety/security, energy efficient, and more reliable vehicle. The quest to provide fuel-efficient, environmental friendly vehicles and the concern of safety/security are
becoming an everyday concern for consumers not only in the automotive market but in our daily lives. Also these are the problems this study is focusing on.

Along with the flood of breakthroughs and innovations in the world of automotive technology, there has been considerable attention given to the most crucial element of environment and driving. Automobile exhaust emissions contribute about 10% of the world’s air pollution problems with carbon monoxide and nitrogen oxide emissions. The increase in automobile fuel consumption threatens the world’s oil reserves where considerable part of allocation is dedicated to transportation. As environmental concerns mount, governmental regulations are being driven towards alleviating these problems. Engine controls can meet stricter emission laws and fuel economy standards. Power train computers adjust the engine and transmission for best performance. The electronic content in engine controls creates a networked, closed-loop system that can manage the emissions and the fuel economy of the vehicle by creating the perfect ratio of fuel/air mixture.

Although there has come to be a vehicle flourish along with the flood of breakthroughs and innovations in the world of automotive technology, by far many traffic accidents still happen here and there. According to World Health Organization figures, an estimated 1.17 million deaths occur and over 10 million people are crippled or injured worldwide each year due to road accidents. The safety/security processors remind you to use seat belts, warn you of hazards, and deploy air bags during an accident. Automotive security and safety takes place even before and when a journey begins. It includes the development in the areas of active safety technology, which is allowing us to actively predict the occurrence of traffic accident, and passive safety technology, which allows us to reduce injury to persons involved in any accident that does happen. Also it includes the self security of automotive, preventing the automotive from being stolen and robbed.

In the field of vehicle safety and security, a major trend sweeping the automotive industry is the transition from mechanical connections to fault-tolerant electric/electronic systems using wires, controllers, sensors and actuators to control mechanical functions such as steering, braking, throttle and suspension. This technology connects the entire automotive to a unit system combining different embedded parts.

1.3 The Organization of This Dissertation

This dissertation is organized as 6 chapters. Chapter 1 introduces the research background, motivation and objective. The research concerns on automotive embedded system design. Embedded design is one of the promising technologies, while automotive control design for less environment pollution and more safety and security, is a key point for modern society. Applying the advanced technique for the society-concerned objective is a very meaningful research topic.
Chapter 2 describes the key points of the current research which mainly focuses on embedded system design for automotive controls, including engine intake control system, anti-lock brake system and automotive security system. In this chapter, some relative technologies of these systems are briefly introduced. Compared to existing techniques, our design method using embedded technique is illustrated, and respectively the three systems are generally described.

Chapter 3 proposes the hybrid embedded design of engine intake control system. First, the mathematical model of hybrid system is presented. Based on the model, the hybrid simulation algorithm is developed. After introducing the basic definition, the proposed algorithm is explained in detail, including mechanism and representation of event, event detection and location, integration formula on the sliding surface, and extended hybrid automaton. Furthermore, this method is applied for the intake system of turbocharged engine and the intake system model and its simulation using the proposed method are also described in detail. Some simulation results are shown to evaluate the efficiency and validity of our proposed method. Finally, the intake system implementation is introduced.

Chapter 4 proposes a general virtual vehicle system and the anti-lock brake system (ABS) control prototype based on the system. The virtual system consists of pure software simulation platform and hardware in loop (HIL) simulation platform. On the first platform designers can run simulation of the new ABS controller in conjunction with the rest vehicle models to study the behavior of the overall system and to optimize the algorithm and logic used in it before building any prototypes. In the second part resulting prototypes are validated in HIL simulation that includes the effects of the hardware-vehicle interaction, actual vehicle components, ABS controller and actuator. Using the developed platform, based on the fuzzy logic control method, an ABS control prototype is also proposed.

Chapter 5 proposes a security system based on the embedded techniques for automotive security. The proposed method for this system includes feature points extraction from shot images and human motion recognition and match analysis. The algorithms are based on independent component analysis (ICA) and cluster algorithm, which are also introduced in this chapter. On the base of the two algorithms, the system is designed using the method of hardware and software co-design to be implemented. The simulation results of the designed system are also presented.

Finally, chapter 6 provides conclusions of this research. The fruits in the field of automotive engine, ABS and safety system are briefly reported.
Chapter 2 Points of Current Research

2.1 Embedded System Design for Engine Control System

In the modern society, as the rapid development of automotive industries, environment pollution gradually becomes a challenged problem to which more and more people pay much attention. The increased environment awareness and requirement for drivability have raised the interest and investment in the researches of complicated automotive modeling and control methods. One of the researches concerns on a high power output while still maintaining a good fuel economy. It can be achieved using a smaller but turbocharged spark ignited engine with a three way catalyst to reduce emissions.

Actually, the task is mostly taken by enhancing the air-fuel ratio control. The air-fuel ratio is the ratio of air-mass and fuel-mass in the cylinder when the valves are closed. The mass of air flowing inside the cylinder can be achieved from the pressure in the intake manifold and the cylinder air charge efficiency. The engine control unit (ECU) tries to get the air pressure of intake manifold and estimate the cylinder air charge efficiency, based on which it can decide the mass of fuel to inject. Thus it is natural to focus on the air path where there are differences. In addition, it has been argued that the air dynamics has a more significant influence on the air-fuel ratio than the fuel dynamics (Powell et al., 1998b). Hence, a key component for precise air-fuel ratio control is the achievement of precise intake manifold pressure or mass flow. Transient air and fuel estimation are still difficult tasks since there are considerable non-linear dynamics between actuators and sensors on the engine. Therefore, in the case that the air path has been a thoroughly studied topic for naturally aspirated engines, additional research on the air system of turbocharged engine is continued because of its more complex intake system, in which there are couplings between the intake and exhaust side that influence the intake manifold pressure and the cylinder air charge efficiency.

Generally the mathematical model of engine intake system to calibrate the air pressure inside it is developed and this model is used for real time predictive control. The model of engine intake system is very complicated, and furthermore when the turbo charger is equipped it becomes more complex. Even though the precise model is developed, the calculation method is not suitable enough for predictive control in real time because of the solution speed limitation of current hardware and software system. For
example, the emissions of hydrocarbons and carbon monoxide are reduced if the injection is finished around intake valve opening. See e.g. (Bouza and Caserta, 2003). This means that the fuel is injected before the induction stroke starts. Therefore in transient conditions, the ECU has to predict the mass of air in the cylinder before intake valve opening. The required prediction time is at least the sum of the computation time, the injection time and the delays of actors. Typically, the necessary prediction time is around one revolution and because modern engine is a machine with high rotation speed, the time of one revolution means the level of millisecond.

Hybrid modeling is a good way to describe automotive engine system because the intake system of turbocharged engine has the nature characteristics of hybrid. The states of air pressure and flow mass can be expressed by continuous state variables; the control commands of throttle plate angle inputs and influence of turbo charge can be expressed by discontinuous variables. Under these constrains, the air flow in the intake system will have the characteristics of acceleration, deceleration even reverse. Modeling and solving this system from the aspects of hybrid system reflects the essence of the system and is closest to the physical realities.

However, since the class of hybrid control problems is extremely broad (it contains continuous control problems as well as discrete event control problems as special cases), it is very difficult to devise a general yet effective strategy to solve them. In our opinion, it is important to address significant application domains to develop further understanding of the implications of the hybrid model on simulation algorithms and to evaluate whether using this formalism can be of substantial help in solving complex, real time control problems. Furthermore, almost all the control algorithms are implemented on microcomputer units which interact to practical plants. As computing tasks performed by embedded devices become more sophisticated and the need for speed and stability of embedded software becomes more apparent. We are facing the problem of how to get the most precise trajectory of system by the least cost. This means the faster and more stable software have to be developed for practical plants on the situation of current hardware limitation.

A novel hybrid simulation algorithm is developed and this algorithm is implemented to solve the model of intake system of turbocharged engine for predictive control. The parameters outputted by this model are the most important parameters in engine control system. Hybrid system is a non-smooth system consisting of sets of differential equations and discrete variables according to the external control commands and internal evolution rules. In this case the system isn’t suitable for direct numerical methods since it has character of chatting (oscillation) between the intersections of different regions in certain situations. A first issue of great practical importance in the procedure of hybrid system simulation is whether the solver can detect the event precisely. In the proposed approach
the sign of the event function is monitored and an event is searched in the span of an integration step, from an approximation of current state variable at current time to the approximation of the next step state variable at next time by fixed step size. When the sign of multiply of event function changes from positive to negative or from negative to positive, an event happens and the system trajectories cross the switching surface. A first-in-first-out stack is used to store the calculated approximation of variable states. The events are discriminated as basic events and induced events which are affiliate to the basic ones and will not trigger the location change if the basic events are not trigged. The processing makes it relatively straightforward to implement numerical algorithm and reduces the number of checks that have to be made every time when the event is triggered. Therefore the program is more efficient and faster. In the procedure of event location, the transition phenomena are analyzed around the switching surface and design the integration formula based on Filippov structure to calculate the integration on the sliding surface. According to the event mechanism and integration formulas, an additional node in the procedure of simulation is added as the extension of the common hybrid automaton. The calculation algorithm is presented to transit the system near the switching surface of two regions into three nodes therefore the undesirable transitions are avoided and the solutions are smooth and efficient.

The model of intake system of turbocharged engine is built from the analysis of thermodynamic and hydrodynamic characteristics and sampled experiment data. The model is embedded into the engine control unit to estimate the air mass flowing into the cylinders. The current parameters are sampled by the sensors and the next step values are calculated by the embedded internal model. According to these values and the compensation parameters such as water temperature, engine rotation and etc., the fuel injection can be decided. Therefore the calculation speed should be fast enough to satisfy the requirement of engine rotations. This model is expressed by a set of differential equations with condition selection on the right hand side and it is developed based on the view of hybrid system and solved using the propose algorithm. The trajectories are smooth under the entire regions of throttle angle inputs. Furthermore the calculation speed is improved at least eight times against to former method and the error is restricted to be less than 1%. This solution is verified on the platform of MATLAB and Visual C++. At last the intake system model is implemented on FPGA chip and it can be embedded into ECU for real time control.

2.2 Embedded System Design for ABS and Development Tools

Vehicle safety continues to be a critical issue at all levels of the automotive supply chain. And, as consumers spend more and more time on the road they are constantly looking for
safety improvements. There are two types of safety system existing. Active safety aims at preventing accidents happening in the first place. Cars are equipped with a number of systems to help the driver control the car before an accident might occur. Passive safety describes the safety systems which are built into cars to protect the driver, the occupants and other vulnerable road users after the accident has happened. Developments in active safety offer real life saving advantages to drivers, particularly in the wet, winter months. With the facts before them, it is believed that drivers would unhesitatingly demand these systems in their cars as they offer substantial benefits in reducing accidents on our roads.

Anti-lock brake system (ABS) is one of the most important equipments in the active safety system. It is a system on motor vehicles which prevents the wheels from locking while braking. The purpose of this is twofold: to allow the driver to maintain steering control and to shorten braking distances by allowing the driver to fully hit the brake without the fear for skidding or the loss of control.

On high-traction surfaces such as bitumen, whether wet or dry, most ABS-equipped cars are able to attain braking distances better (i.e. shorter) than those that would be easily possible without the benefit of ABS. For a majority of drivers, in most conditions, in typical states of alertness, ABS will reduce their chances of crashing, and/or the severity of impact. In such situations, ABS will significantly reduce the chances of a skid and subsequent loss of control. In gravel and snow, ABS tends to increase braking distances. On these surfaces, locked wheels dig in and stop the vehicle more quickly. ABS prevents this from occurring. Some ABS calibrations reduce this problem by slowing the cycling time, thus letting the wheels repeatedly briefly lock and unlock. The primary benefit of ABS on such surfaces is to increase the ability of the driver to maintain control of the car rather than go into a skid— though loss of control remains more likely on soft surfaces like gravel or slippery surfaces like snow or ice.

A typical ABS is composed of a central electronic unit, four speed sensors for four wheels respectively, and two or more hydraulic valves on the brake circuits. It is a complicate system integrating mechanics, electronics and control devices, and the development procedure has also change greatly against the old methods. This process can be divided into 5 basic stages:

1. The first stage consists of system definition where design engineers specify and define the requirements for the embedded control system. This is often done with text based files created on a desktop PC. In some instances real-world empirical data is acquired as part of the specifications.

2. In the second stage, rapid prototyping, the design engineer develops the control strategy in a simulated environment on the desktop PC or workstation and then creates an initial prototype of the system with real-time hardware.
3. In the third stage, the code generation phase, production code is generated and manually tweaked for the target hardware. At this point the production code is running on the target hardware and not on the prototype hardware anymore.

4. During the fourth stage, the design engineer will test out the target hardware against a simulated environment. Real-Time hardware is used to simulate the real-world environment that the control system interfaces with.

5. Finally, the target hardware is deployed and integrated into the system and final testing is done to ensure that design specifications were met.

Within all five stages, computer simulation plays a vital role. Computer simulation for vehicle components design is progressing greatly in recent years. During computer simulation, computer models are used to recreate or simulate the vehicle environment, and the ECU is then interfaced to the simulated environment. It is only in recent years that the virtual simulation of full vehicle systems has become a serious effort for the automotive industry. Most of the large companies have developed their own facilities to simulate vehicles in virtual realistic environments, each of which is in a different setting with different research objectives.

A general virtual vehicle system is developed and made specified modification for ABS controller design, furthermore the ABS control prototype is built based on this system. This system consists of two connected parts: pure software simulation and hardware in loop (HIL) simulation. In the first part all the components are modeled in software, developed in the platform of MATLAB/SIMULINK. ABS control logic is developed and tested in conjunction with the vehicle models to study the behavior of the overall system and to optimize the algorithm used in it before building prototypes.

In the second part HIL platform is constructed including the computer cluster, the hardware-vehicle interaction (sampling, time lags, etc.), actual vehicle components, ABS controller and actuator. All the components are connected together by controller area network (CAN) (SAE, 2056/1, and 2056/2, 1994), including engine, ABS controller, sensors and vehicle model. Rather than testing these components in complete actual system setups, virtual system allows the testing of new components and prototypes by communicating with software models on the main computer by CAN interface. Furthermore this technology is flexible enough to allow expansion and reconfiguration, in accordance with the development of modern automotives. For the requirement of real time system, one computer is used to run the vehicle model exclusively and use another one for data and graphic processing; they are connected through Ethernet based on TCP/IP. In the procedure of HIL simulation, a novel simulation algorithm is proposed to deal with the abrupt changes of hydraulic pressure and make the entire system robust and stable. The structure of entire hardware and software system is shown in Figure 2.1:
Figure 2.1 Entire structure of virtual vehicle system.

An integrated user-friendly interface including vehicle parameters database editors, configurations and visualization tools are also used to interact with the core components. MATLAB, database and EXCEL are integrated into this system through Visual C++ programming platform on assistant computer. Microsoft ACCESS is chosen as the data storage database and ADO is used for data operation. The experiment data is stored into database through Ethernet. The necessary data can be taken out from database and processed in MATLAB after being converted to the proper data format. Furthermore the desirable data and figures can be imported into EXCEL, which convenience the data analysis and exportation for ABS designers.

Conventional ABS control algorithms must account for non-linearity in brake torque due to temperature variation and dynamics of brake fluid viscosity. Although fuzzy logic is rigorously structured in mathematics, one advantage is the ability to describe systems linguistically through rule statements. The superior characteristics through the use of fuzzy logic based control are realized rather than traditional control algorithms to ABS controller. Due to the nature of fuzzy logic, influential dynamic factors are accounted for in a rule based description of ABS. This type of intelligent algorithm allows for improvement and optimization of control result. This algorithm is tested on the HIL platform and the desirable results are achieved.

### 2.3 Embedded System Design for Automotive Security

With increasingly sophisticated security devices reaching the automotive market, one
would think auto security has come to be in business. The events of disasters for vehicles and drivers happen frequently so that more and more devices for a vehicle security effectively appear. On one hand, the car security trade is alive and well. On the other hand, for the automotive electronics designers, the automotive industry’s primary goals of safety, cost, and reliability have to be concentrated on. Therefore, embedded technology are more and more used to design automotive systems so that many embedded products for security come into the market. The devices include digital devices, password devices, communication devices, video system and so on. Among the digital devices for automotive safe, digital key with the embedded small wireless radio sender, the corresponding receiver is installed inside the car. When the key is inserted, the wireless signal with twenty digital codes is sent, if they are ensured, the car will run. There is also a password lock. This lock is based on the principle of radar radio. There is a chip inside the key hole. When the key turns, the chip will check the password sequence. The communication system for a car safe is also popular. For example, a telephone control system controls cars by calling such as closing door and windows, controlling fuel supply and closing other electronic devices. A GPS vehicle satellite navigation system tracks cars by consulting the geography system and avoiding electric wave interference. Moreover, recently video systems for car safe have come to be developed. A micro-spy camera is one of them. Its size is so small that it can be located at any place of cars. The camera can also work under the weak light and be connected to the phone networks. The shot images will be transferred to the control center to be recognized.

Those devices and systems are helpful to vehicle safety, but they also have some defects. The digital device only focuses on the key. If the key is lost, the device has no use. Although the communication system can track the objects, this kind of monitor costs too much and is very expensive. The camera is useful to monitor but only depending on the camera is too simple because the persons’ behaviors have to be judged.

Among security systems in the field of automotive and others, tracking, recognizing and detecting objects using a video sequence are topics of significant interest. Cai and Agganval (Aggarwal and Cai, 1999) reported a variety of methods for analyzing human motion. In (Harwood and Davis, 1998; Haritaoglu and Davis, 1999), Haritaoglu, Hanwood and Davis tracked both single and multiple humans in outdoor image sequences using a PC. In (Oliver et al., 1999), Nuria, Rosario and Pentland recognized two-person interactions from perspective-view image sequences. They tracked persons and recognized their interaction using trajectory patterns. Meanwhile, some researchers developed applications include one or more embedded systems. Especially recently, more and more embedded systems are employed for security systems. In (Pentland, 2000), Pentland proposed a “wearable device” that sees people using an image sensor and understands the environment. In such equipment, a computer can act or respond
appropriately without detailed instructions from humans. In (Mahonen, 1999), Mahonen proposed a wireless intelligent surveillance camera system that consists of a digital camera, a general-purpose processor or DSP for image processing and a wireless radio modem. In (Shirai et al., 1999), Shirai introduced a real-time surveillance system with parallel DSP processors (TI TMS32OC40). Their system consists of several boards connected in series. It can compute optical flow computation in floating point faster than 30 frames per second. In this system, a DSP is located between the two memories. The DSP computes and transfers the image data from the video memory to the other memory, which is connected to the next processing stage.

A security system is built based on Independent Component Analysis (ICA) for automotive security. The goal of ICA is to recover independent sources given only sensor observations that are unknown linear mixtures of the unobserved independent source signals. In contrast to correlation-based transformations such as Principal Component Analysis (PCA), ICA not only de-correlates the signals (2nd-order statistics) but also reduces higher-order statistical dependencies, attempting to make the signals as independent as possible. This character is well used in the field of image recognition.

Unlike other methods, the proposed method is to detect the abnormal motion from activities of a person himself based on ICA. In the first step almost images are defined. These images are about abnormal motions of people around the door of automotive according to the opinions of observers. These motions are stored in image database, which is used for motion recognition through matching the real time image caught by a micro camera. Before ICA processing, the image sizes are normalized, both the shot images and the images in the database. In order to adapt to different type of camera and color content, each pixel (RGB-triple) is projected onto a plane by average RGB values. The feature vectors of images are extracted using ICA and organized into categories in order to improve the precision and decrease time consumption using cluster algorithm. In the procedure of abnormal pattern recognition, the basic cluster is specified in advance and make the program identify the remained images into basic cluster automatically. A novel similarity calculation method is developed to recognize the most similar image from the certain cluster according to the extracted feature vectors. The array of feature vectors and the pattern matrix of images in the database extracted using ICA are stored in RAM on the FPGA board and therefore the image database is seamlessly integrated to the real time system.

In the second step, the micro digital camera is embedded into a terminal board and fixed in the rearview mirror of car. It is used to catch the movements of people who present around the door kept under surveillance and do some preprocessing. The image data is transferred from the sending module of Ni3, which is a wireless communication device to the receiving module, connected to the FPGA board for receiving data. The
feature vectors of shot images are extracted using pattern matrix. From comparing the feature vectors of shot image and the cluster centroid, it is possible to get which cluster the shot images belong to respectively. After that the most similar image can be achieved from the corresponding cluster. The security level of a person can be calculated and any appearance of a person deemed threatening can be set to trigger an alarm. Furthermore this system can be connected to engine management system to control the start of engine. Therefore if a person is regarded as insecurity, he can not run the can even if he enters into the cab.

In designing the security system, it is considered to take full advantage the unique characteristics of the embedded system. For example, image processing such as feature extraction and cluster is more effectively performed on an embedded processor. The computational complexity of each operation and the transfer rate and overall suitability of each processor were evaluated. In the discussion below, the algorithms, calculating stages and simulation results are described. Figure 2.2 presents an overview of the system.

![Flow chart of security system](image)

**Figure 2.2 The flow chart of security system.**
Chapter 3   Hybrid Embedded Design of Engine Intake System

3.1 Mathematical Model of Hybrid System

The earliest direct reference of hybrid system, as it is known, is the visionary work from MIT, which formulated a class of hybrid-state, continuous-time dynamic systems and examined an optimal control problem (Witsenhausen, 1966; Mohrenschildt, 2000; Bemporad and Giorgetti, 2003). Since then hybrid system appears both in automatic control (Brockett, 1993; Gennaro et al., 1994; Gollu et al., 1989) and in computer science (Anstaklis et al., 1995; Bobrow and Arbib, 1774). Depending on the researcher’s background and the actual problem they can use timed automata, dynamical systems theory, automata theory, discrete event systems, programming verification methods, logic programming etc to define and analyze hybrid system. Now hybrid system has been used successfully to address problems in air traffic control (Tomlin et al., 1998), automotive control (Balluchi et al., 2000), bioengineering (Brockett, 1996), process control (Lennartsson et al., 1996; Engell et al., 2000), highway systems (Horowitz and Varaiya, 2000), and manufacturing (Pepyne and Cassandras, 2000). The needs of these applications have fuelled the development of theoretical and computational tools for modeling, simulation, analysis, verification, and controller synthesis for hybrid systems (Tomlin et. al, 2003; Franceschet et. al, 2003; Qin and Jia, 2002).

In general a hybrid system can be in one of several modes of operation, whereby in each mode the behavior of the system can be described by a set of difference or differential equations, and that the system switches from one mode to another due to the occurrence of events. The mode transitions may be caused by an external control signal, or by an internal control signal if the controller is already included in the system under consideration. Also these transitions can be caused by the dynamics of the system itself, i.e. when a certain condition is satisfied, or a certain boundary in the state space is crossed. It is expressed by guards, as long as the guard conditions of a given node are not satisfied the system stays in that mode. At a switching time instant there may be a reset of the state (i.e. a jump in the values of the state variables) and/or the dimension of the state may change.
The definition of hybrid automaton (Sava and Alla, 2001; Johansson et. al, 2000) is illustrated using a simple thermostat model given in Figure 3.1. The thermostat model consists of three locations, that is \( L = \{ \text{heat}, \text{cool}, \text{check} \} \). It contains two continuous variables, namely a clock \( t \in \mathbb{R}_{\geq 0} \) and a temperature \( T \in \mathbb{R}_{\geq 0} \). In this particular example the continuous state-space can be limited such that both the clock \( t \) and the temperature \( T \) are within the interval \([0,100]\) without loss of generality of the analysis. The continuous state thus is \((t, T) \in [0,100]^2\). A state is denoted with \((heat, 2, 8)\) representing \(t=2 \land T=8\) while in location heat. The continuous dynamics of the clock \( t \) is \( i = 1 \) in all locations. The thermostat is switched on in the heat location, so that the temperature increases by \( \dot{t} = 2 \). The invariant in the heat location is \( T \leq 10 \land t \leq 3 \), that is,

\[
I(heat) = \{(t, T) \in [0,100]^2 | T \leq 10 \land t \leq 3\} \tag{3.1}
\]

The thermostat system, therefore, cannot remain in the heat location when the temperature exceeds ten or the clock exceeds three time units. The control can switch to the cool location, which models that the thermostat is switched off, when the guard \( T \geq 9 \) is enabled. One of the guard sets \( g \) of this transition therefore is

\[
g = \{(t, T) \in [0,100]^2 | t \leq 3 \land 9 \leq T \leq 10\} \tag{3.2}
\]

This means, the switch from the heat location to the cool location can happen non-deterministically at any time when the temperature \( T \) is in the interval \([9,10]\). The control remains in the cool location, until the temperature is in the interval \([5,6]\)

Figure 3.1 A simple hybrid system model of thermostat.
according to the guard conditions, when it switches back to the \textit{heat} location. This transition has a reset, which resets the clock $t = 0$. The third location \textit{check}, models a self-checking mode of the thermostat controller. The invariant in the \textit{check} location guarantees that the control will return to the \textit{heat} location after at most one time unit. During this time, the temperature drops, but this happens slower than in the \textit{cool} location. It is assumed that initially the thermostat is in its \textit{heat} location with $t = 0$ and $5 \leq T \leq 10$.

In the term of control and system theory and the application of practical engineering, the problems which are paid more attention to are: What mix of continuous and discrete properties is rich enough to capture the properties of the system that is modeled? How can it be verified that the hybrid model satisfies the demands on performance and stability in practice? What are focused on are the hybrid systems of the interaction between continuous-variable systems (i.e. systems that can be described by a system of difference or differential equations) and discrete-event systems (i.e. asynchronous systems where the state transitions are initiated by events).

The process of identifying generic patterns for modeling and implementing hybrid system begins by reviewing a large number of recent publications and engineering experiments. These papers ranged from the theoretical hybrid modeling techniques to the applications of hybrid control to the real world system (Antsaklis and Nerode, 1998; Branicky et al., 1998; Fierro et al., 1999; Frazzoli et al., 1999; Garcia et al., 1995; Koutsoukos, 2000; Liu et al., 1999).

Mathematical models are frequently used in many disciplines of science to study complex behavior of system. System that can be modeled by differential equations is called dynamical system. A dynamical system starting from a particular initial state can evolve towards a steady state or to irregular chaotic motion. A hybrid automaton is a dynamical system that describes the evolution in time of the valuations of a set of discrete and continuous variables.

**Definition 3.1 Hybrid Automaton:**

Hybrid automaton can be defined as tuple: $H = (Q, X, U, Y, M_c, M_d, E, R)$ or respectively $H = (Q, X, U, Y, M_d, M_c, E, R)$ where

\begin{itemize}
  \item $Q$ is a finite collection of discrete state variables taking values in the set of discrete states $Q = \{q_1, q_2, \ldots, q_n\}$, it is also called locations, modes or nodes;
  \item $X$ is a finite collection of continuous state variables taking values in the continuous state space $X = \mathbb{R}^n$;
  \item $U$ is a finite collection of input variable contains discrete variables in $U_d$ and continuous variables in $U_c$;
  \item $Y$ is a finite collection of continuous outputs variables, taking values in the set $\mathbb{R}^p$;
  \item $M_c$ is the class of time continuous dynamical system defined by the equations:
\[
\dot{x}(t) = f_i(x(t), u(t)) \\
y(t) = h_i(x(t), u(t)) \quad i \in J
\] (3.3)

where

\[ t \in \mathbb{R}, \quad j = 1, 2, \ldots, n \subset \mathbb{N}, \quad \text{each } f_i \text{ is globally Lipschitz continuous. Or } M_d \text{ is the class of discrete time dynamics defined by the equations:}
\]

\[
\dot{x}(t + 1) = f_i(x(t), u(t)) \\
y(t) = h_i(x(t), u(t)) \quad i \in J
\] (3.4)

where

\[ t \in \mathbb{N}, \quad j = 1, 2, \ldots, n \subset \mathbb{N} \text{ and } f_i \text{ is a discrete expression. The solution } x(t) \text{ in both of the two forms exits and unique;}
\]

\[- M \quad Q \rightarrow M_d \text{ or } Q \rightarrow M_d \text{ is a mapping associating to each discrete state a continuous time or discrete time dynamical system;}
\]

\[- E \subset Q \times U \times Q \text{ is a collection of discrete transactions;}
\]

\[- R \quad E \times X \rightarrow X \text{ assigns to each transaction a reset function.}
\]

A hybrid automaton basically combines two paradigms of state space models described by continuous dynamics as well as discrete transitions. Each discrete state is called a location as defined by \( Q \). Associated with each location are the continuous dynamics described by a differential equations or the discrete dynamics described by the difference equations. In each location the continuous dynamics and discrete dynamics evolve according to their own inclusions. The hybrid automaton may reside in the current location as long as the states remain inside its invariant. The discrete transitions \( E \) between locations are labeled with the guard and reset conditions. Each discrete transition may be taken when the guard conditions are triggered and the reset condition must be satisfied after the discrete transition is taken.

Many engineering systems are best described by sets of ordinary differential equations (ODEs) with discontinuous right-hand sides. Such systems arise in many contexts and are often referred to as hybrid switch system in the fields of control theory by mathematical abstraction. Examples include phase transitions, contact mechanics, or the dynamics of physical systems controlled by digital computers. From engineering view, they can also be regarded as discrete event systems augmented with differential equations. The most basic form of such a system is:

\[
\dot{x}(t) = \begin{cases} f_i(x(t), u(t)) & x \in S_i \\ f_j(x(t), u(t)) & x \in S_j \end{cases}
\] (3.5)
With the initial condition \( x(0) = x_0 \), where \( x(t) \in \mathbb{R}^n \) is \( n \)-dimensional state vector and \( f(x(t), u(t)) \) is the vector fields of right hand side describing the time derivative of the state vector. If the input \( u(t) \) is integrated into the system, it is also possible to describe the differential equations as \( f(t, x(t)) \). A dot \( (\cdot) \) denotes differentiation with respect to time. All the unnecessary parameters are omitted for convenience. Here only the continuous time system is considered and it can also be extended to discrete time system.

The transition of differential equations can be divided into two types according to their degree of discontinuity:

1. System exposes discontinuous jump in the state, like impacting system with velocity reversal, such as a bouncing ball. This situation causes the transaction on the state variables, not the locations. Or in other words, it is sometimes a self-location transition. This situation is not considered in this thesis.

2. Vector fields are discontinuous without state variables jump. The physical model is abstracted to this type of system and this one is mainly treated. This is a transition among different locations. There are generally two situations. First, non-smooth continuous system with a discontinuous Jacobian, those system are described by a continuous vector field but the vector field is non-smooth. Second, systems described by differential equations with a discontinuous right-hand side.

The consideration is restricted to the differential equations with the right-hand sides that are discontinuous of non-smooth on a single switching surface \( S_y \). In this term what defined in automaton is integrated through synthesizing all the variables into the equations. The state variables staying in the regions of \( S_i \) and \( S_j \) correspond to the trajectories evolving in the locations of \( q_i \) and \( q_j \). The switching surface is defined by the scalar indicator event function \( s_y(x(t), u(t)) = 0 \). The description of hybrid system in engineering terms is consistent to the mathematical model, but much straighter to be understand and nearer to practical applications (Clarke et al. 1998). It is a \( n \)-dimensional nonlinear system that the right-hand side \( f(x(t), u(t)) \) is assumed to be discontinuous but it is piecewise continuous and smooth on \( S_i \) and \( S_j \). The function \( f_i(x(t), u(t)) \) is assumed to be on \( S_i \cup S_y \) and \( f_j(x(t), u(t)) \) is assumed to be on \( S_j \cup S_y \). It is not required that \( f_i(x(t), u(t)) \) and \( f_j(x(t), u(t)) \) agree on \( S_y \). The state space \( \mathbb{R}^n \) near \( s_y \) is split into two regions \( S_i \) and \( S_j \) by switching surface \( s_y \). The switching surface can be generated from autonomous switching or controlled switching rules that can be unified in \( s_y \) if necessary. The trajectory vector is on \( S_y \) when \( s_y(x(t), u(t)) = 0 \). The switching surface is corresponding to the guard conditions in the theoretical definition of automaton. Here two regions are extended with one switching surface from hybrid system mathematical model, more general polynomial forms can be treated by the same techniques at the expense of rather more complicated mechanism.
3.2 Preliminary of Hybrid Simulation Algorithm

3.2.1 Concepts and Structures of Hybrid Simulation
The goal of the solution algorithm presented here is to address a novel numerical simulation technique that properly deals with the unique features of hybrid switching system. Our object is to solve the practical problems efficiently and stably as well as simulate their behavior model more accurately by exploiting their features. Even though techniques for dealing with some other peculiarities of hybrid system have been introduced by other communities in different contexts, such as distributed discrete event system simulation, they are not efficient and stable enough to satisfy our requirements.

Traditional approaches to the modeling of hybrid system concealed the integration of continuous and discrete nature of the systems by converting them into either purely discrete or purely continuous systems. These approaches neglect the nature property of hybrid system and there exist considerable errors in the simulation results. There are many numerical solution methods and software for ODEs, but these methods mainly focus on the continuous aspects and cannot be directly applied to hybrid system. In practical applications the hybrid model is usually simulated in one of the modes, the switching points are determined and then the system is simulated in the next mode. For the active control of hybrid system, however, this mode-by-mode approach is typically not applicable, since the evolution and control may influence the switching and, hence, the hybrid system has to be considered as a whole.

On the other hand, many papers that employ some sort of switched control for practical systems are concerned with the transition. One approach pursued by Oishi and Tomlin (Oishi and Tomlin, 1999) creates a new discrete state for the transition that incorporates the transition dynamics into it. This approach can create a large number of extra transition states if the original hybrid system has a large number of discrete states originally, considering all the combinations of discrete states and the corresponding transitions between them. Another approach to handling transition dynamics is to find a means of smoothing the transition between discrete states without deviating from the original set of discrete states. A common method of achieving this goal is to smooth the control action. For example, in many gain-scheduled control algorithms, controller parameters are switched based on the state’s inclusion in regions about local operating points (Nichols and Reichert, 1993; Jeon, 2001). When the state nears the boundary of two regions, the parameters are blended to smooth the transition from one region to the next. Another example of control smoothing is used regularly in sliding mode control, which is a switching control law where a switching surface is defined in the state space. To reduce chatter, the discontinuous part of the control is smoothed in a region around the switching surface. These methods can simulate hybrid system more precisely but
unfortunately they are expensive to compute the switching condition and do much on the switching surface. Therefore these methods are not suitable in real time control systems.

Simulation and computational capability of hybrid system is a step towards exploring the characterization of the corresponding physical models. In terms of the types of system they can be described by or implemented with continuous differential equations and discrete transitions. Therefore simulation of hybrid system consist two parts: continuous part and discrete part. The basic simulation algorithm is shown in Figure 3.2. It is demonstrated that the proper way of simulate any hybrid system during the continuous evolution is to numerically integrate all of the differential equations with holding the discrete locations constant. The continuous integration should stop and turn to discrete processing on the time at which the final state \( s(t_N) \) generated in the sequence is as close as possible to \( t \) -the first time making \( s(t, x(t_N)) = 0 \). Discrete processing includes autonomous switching and controlled switching. In the procedure of switching, precisely finding and locating the hybrid time trajectory \( t \) and do the corresponding processing are called event detection and location. Often this is done in two phases. The first phase occurs at every integration step and consists of determining if \( \exists t \in I_i \) that \( s(t, x(t_N)) = 0 \). This is called the event detection phase. If such a time exists, the event location phase is activated in which a more precise computation is need. After a transition happens, the discrete component of the state is updated to another location. Once no further discrete evolution can occur, the integration may be restarted with \( (t, x(t_N)) \) as the new set of initial conditions.

Figure 3.2 High level flow chat of algorithm.
The transition events for autonomous switching are processed as follows. Each differential equation of vector fields output a current state, the states are transferred to event function to produce a signal of event happening. Therefore a discrete signal is generated from the states input according which the updated vector field can be decided and a new loop of integration will begin. The autonomous switching is more complicated than the controlled switching in the procedure of simulation so this study is more focusing on this aspect.

3.2.2 Origins and Definitions of the Algorithm

The physical system of \( f(t, x) \) can be modeled by differential equations. If the vector field is smooth, that means \( f(t, x) \) is continuously differentiable up to any order both in \( x \) and \( t \), then the solution \( x(t) \) of this system exists for any given initial condition. However differential equations stemming from hybrid system are discontinuous, i.e. the right hand side of \( f(t, x) \) can be discontinuous in \( x \). The theory of Filippov (Filippov, 1964, 1988; Sastry, 1999) gives a generalized definition of the differential equations which incorporates systems with a discontinuous right hand side.

In order to make things be as clear as possible, first look at a very simple one-dimensional example (Kunze and Kupper, 1997). Consider the following differential equation with a discontinuous right-hand side:

\[
\dot{x} = f(x) = 1 - 2 \text{sgn}(x) = \begin{cases} 
3 & x < 0 \\
1 & x = 0 \\
-1 & x > 0 
\end{cases} 
\] (3.6)

with \( \text{sgn}(0) = 0 \).

For a general given initial condition \( x(0) \neq 0 \) a solution of the initial value problem can be obtained.

\[
x(t) = \begin{cases} 
3t + c_1 & x < 0 \\
-t + c_2 & x > 0 
\end{cases} 
\] (3.7)

with constants \( c_1 \) and \( c_2 \) being determined by the initial conditions. From the equation (3.7), each solution will reach \( x = 0 \) in finite time. If the solution arrives at \( x = 0 \), it can not leave \( x = 0 \), because \( \dot{x} > 0 \) for \( x < 0 \) and \( \dot{x} < 0 \) for \( x > 0 \). The solution will therefore stay at \( x = 0 \), which implies \( \dot{x}(t) = 0 \). Note that \( x(t) = 0 \) with \( \dot{x}(t) = 0 \) is not a solution in the classical sense since \( 0 \neq 1 - 2 \text{sgn}(0) \) according to the equation (3.6). The natural idea to extend the notion of solution is to replace the right hand side
$f(x)$ by a set-valued function $F(x)$ such that $F(x) = \{f(x)\}$ if $f$ is continuous in $x$. If $f$ is discontinuous in $x$ a suitable choice of $F(x)$ is required. The differential equation is then replaced by the differential inclusion $\dot{x} \in F(x)$.

Define the set-valued sign function

$$Sgn(x) = \begin{cases} \{-1\} & x < 0 \\ \{-1,1\} & x = 0 \\ \{1\} & x > 0 \end{cases}$$

which is a set-valued at $x = 0$. With this definition this modified system has a unique global solution against to the model that equation (3.6) represents.

Obviously the goal of hybrid system simulation is to produce sufficiently accurate approximations of executions using the least computational effort. Hybrid system modeling and simulation technology are still premature in the mathematical fields and in the practical applications. Before describe the simulation algorithm of hybrid system, some basic theorems and definitions are illustrated. These theorems and definitions reflect the last developments of hybrid system in the research of mathematical aspects. Also they generate some strict and exact mathematical explanations to our simulation algorithm.

For a unique continuous system it is assumed that $f(t,x)$ is linearly bounded, i.e. there exist positive constants $\gamma$ and $c$ such that

$$\|f(t,x)\| \leq \gamma \|x\| + c \quad \forall (t,x)$$

(3.9)

If the vector fields are smooth, which means there is no discontinuity on the right hand side and no jump of state variables, that $f(t,x)$ is continuously differentiable up to any order of $x$.

Hybrid system is constructed by continuous differential equations combining discrete transitions. In every location the dynamic states are described by continuous system, therefore first explain the theorem (Clarke et al. 1998, Sastry, 1999) of the existence and uniqueness of this type of continuous system.

**Theorem 3.1 Existence and Uniqueness of Continuous System**

Suppose $f(t,x)$ that is continuous and let $(t_0, x_0) \in \mathbb{R} \times \mathbb{R}^n$ be given. If we can get the following deductions:

1. There exists a solution of this equation on an open interval $(t_0 - \delta, t_0 + \delta)$ for $\delta > 0$ satisfying $x(t_0) = x_0$.
2. If in addition it is assumed that $f(t,x)$ is linearly bounded, so that inequation 3.9 holds, then there exists a solution of $f(t,x)$ on $(-\infty, +\infty)$ such that $x(t_0) = x_0$. 


3. Let us now add the limitation that $f(t,x)$ is locally Lipschitz, i.e. there exists a constant $L > 0$ such that

$$
\|f(t,x) - f(t,y)\| \leq L\|x - y\| \quad \forall x, y \in \mathbb{R}^n
$$

(3.10)

Then there exists a unique solution of $\dot{x} = f(t,x)$ on $(-\infty, +\infty)$ such that $x(t_0) = x_0$.

This theorem addresses a more and more strict restriction step by step in the definition of the solution of continuous system. Our solution algorithm for continuous part in hybrid system will obey on these restrictions if there is no explicit explanation.

Definition 3.2 Hybrid Time Trajectory

A hybrid time trajectory is a finite or infinite sequence of interval $\tau = \{I_i\}_{i=0}^N$, such that

$$
I_i = [\varepsilon_i, \varepsilon'_i], \text{ for all } i < N;
$$

If $N < \infty$, then either $I_N = [\varepsilon_N, \varepsilon'_N]$ or $I_N = [\varepsilon_N, \varepsilon'_N)$;

$\varepsilon_i \leq \varepsilon'_i = \varepsilon_{i+1}$ for all $i$

The interpretation is that the $\tau_i$ are the times at which discrete transitions take place. Since all hybrid automaton discussed here are time invariant, it is assumed that $\tau_0 = 0$, without loss of generality. Each hybrid time trajectory $\tau$ is linearly ordered by the relation $\prec$ (this symbol defines a time sequence) defined by $t_1 \prec t_2$ for $t_i \in [\varepsilon_i, \varepsilon'_i]$ and $t_2 \in [\varepsilon_j, \varepsilon'_j]$ if $t_1 < t_2$ or $i < j$. It is said that $\tau = \{I_i\}_{i=0}^N$ (discrete of $\tau$) is a prefix of $\tau' = \{I_i\}_{i=0}^M$ and write $\tau \subseteq \tau'$ (this symbol defines span identical) is either they are identical, or $\tau$ is finite, $N \leq M$, $I_i = I_j$ for all $i = 0, \ldots, N - 1$, and $I_N \subseteq J_N$. For a hybrid time trajectory $\tau = \{I_i\}_{i=0}^N$, $\langle \tau \rangle$ is defined as the set $\{i = 0, \ldots, N\}$ if $N$ is finite and as the set $\{0, \ldots, N, \ldots\}$ if $N = \infty$ and $|\tau| = \sum_{\tau_0, \tau_1}^{\tau_{i-1}, \tau_i}$. This definition is a general version of hybrid time trajectory and it is especially appropriate for multi-agent hybrid system, which has each agent a time sequence $\tau$ and has the entire system a time sequence $\tau'$. Each agent of the system is considered to have its own local clock which proceeds at different rate. A part of this definition is only used in our algorithm.

Definition 3.3 Hybrid State Trajectory

It is also called execution in some communities of hybrid research. An execution of a hybrid system is a collection $x = (\tau, q, x)$, where $\tau$ is a hybrid time trajectory, $q: \langle \tau \rangle \rightarrow Q$ is a map, and $x = \{x_i: i \in \langle \tau \rangle\}$ is a collection of differentiable maps $x_i: I_i \rightarrow X$ such that:
\((q(0), x(0)) \in \text{init}, \ \text{init} \) denotes the initial states;

for all \( t \in [\tau_0, \tau_1] \), \( \dot{x}(t) = f(q(t), x(t)) \) and \( x'(t) \in Q(q(t)) \);

for all \( i \in (\tau) \setminus \{N\} \), \( e = (q(i), q(i+1)) \in E \), \( x'(\tau_i) \in G(e) \), and \( x^{i+1}(\tau_{i+1}) \in R(e, x'(\tau_i)) \).

Here \( E \) is transition which is usually defined as edge in the field of computer science and \( G \) is named guard, \( x'(\tau_i) \in G(e) \) is a discrete transition monitor, which means \( x \) is in the discrete state.

It is said that a hybrid automaton \( H \) accepts an execution \( \chi \) if \( \chi \) fulfils the conditions of Definition 3.3. For an execution \( \chi = (\tau, q, x) \), \( (q_0, x_0) = (q(\tau_0), x(\tau_0)) \) is used to denote the initial state. An execution, \( \chi' = (\tau', q', x') \), of \( H \) (write \( \chi \in x' \)), if \( \tau \subseteq \tau' \) and for all \( i \subseteq (\tau) \) and all \( t \in I_i \), \( (q(i), x'(t)) = (q'(i), x'(t)) \). An execution is called finite if \( \tau \) is a finite sequence ending with a compact interval, and infinite if \( \tau \) is either an infinite sequence, or if \( |\tau| = \infty \). An execution is called Zeno if it is infinite but \( |\tau| < \infty \), or equivalently, if it takes an infinite number of discrete transitions in a finite amount of time (Imura and Schaft, 2000). In the following sections these two definitions will be synthesized as hybrid trajectory without explicit explanation.

In our simulation algorithm, the practical terms of simulation results are focused on. The models equations built from the physical system should satisfy the theoretical restrictions and the prototype of simulation can be abstracted from these definitions.

### 3.2.3 Numerical Integration for Continuous Locations

Hybrid system can be addressed as a set of differential equations with discontinuous right hand side. The simulation will begin by approximating the continuous evolution of the state \( x(t) \) described by the flow \( f(\tau, x) \). The state evolution described by the differential equation will be approximated by a numerical integration technique, so instead of the simulator producing a continuous map \( x: \tau \rightarrow X \) (i.e. – an analytical solution to the differential equation), the simulator returns a sequence of approximations of the continuous state at a discrete set of points along the hybrid trajectory \( \{x(t_0), x(t_1), ..., x(t_n)\} \) for \( t_0 < t_1 < ... < t_n \). Therefore numerical integration is a very important aspect in the procedure of hybrid system simulation.

The Taylor series algorithm is one of the earliest algorithms for the approximate solution for initial value problems for ordinary differential equations. The basic idea of the developments the fields of numerical solution of ODEs is the recursive calculation of the coefficients of the Taylor series.

Modern numerical algorithms for the solution of ordinary differential equations are also based on the method of the Taylor series. Each algorithm, such as the Runge-Kutta or the multi-step methods are constructed so that they give an expression depending on a
parameter \( h \) called step size as an approximate solution. The first terms of the Taylor series of this expression must be identical with the terms of the Taylor series of the exact solution. These are the consistency and the order conditions for the algorithms. These expressions potentially can be evaluated at any value of the parameter \( h \), but practically the evaluation is realized only at grid points. Therefore such algorithms give the values of the approximate solution at grid points. Some algorithms differ from others in its order, stability properties and its cost of realization. The overview of the modern algorithms one can find in the monograph of E. Hairer, S. P. Norsett and G. Wanner (Hairer and Wanner, 1987, 1991). A possible implicit extension of the Taylor series algorithm is given in (Molnarka and Raczkevi, 1998). Here on Euler method and Runge-Kutta method, those used in our algorithm are introduced.

Suppose the dynamic system is defined by the continuous ODEs,

\[
\begin{align*}
\dot{x}(t) &= f(t, x(t)) \\
x(a) &= x_0, \ a \leq t \leq b
\end{align*}
\]

where \( f(t, x(t)) : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n \), \( x_0 \) is the initial vector. The instantaneous rate of change of \( x(t) \) at time \( t \) can be calculated in terms of \( x \) and \( t \) alone. Now if \( h \) is any small interval of time it is known that as a first order of approximation of Taylor series it can be written as:

\[
x(t + h) \approx x(t) + h \cdot f(t, x(t))
\]

(3.12)

If \( t_{n+1} = t_n + h \), an estimation \( x_{n+1} \) for \( x(t_{n+1}) \) can be obtained from the estimation \( x_n \) for \( x(t_n) \). Here \( x(t_n) \) is used to denote the precise value of \( x_n \).

\[
x_{n+1} \approx x_n + h \cdot f(t_n, x_n)
\]

(3.13)

Figure 3.3 Forward Euler method.
This technique for finding approximate values for the solution of first order differential equation is the simplest of several similar ones. Each of them proceeds in this stepwise fashion, obtaining an estimate for \( x(t+h) \) from \( x(t) \). This is the forward Euler integration algorithm and it is shown in Figure 3.3 clearly. It is a first order integration method, because it contains the function \( f(t,x) \) evaluated at one point in time. It is also an explicit method, which means that it does not use states or inputs from future values of \( x(t_n) \).

Alternatively, a backward difference is used to approximate the derivative. The backward Euler integration algorithm is a first order implicit method. It is implicit because the state is a function of itself. Implicit algorithms require additional computation to solve for, which often takes the form of an iterative technique such as the Newton-Raphson algorithm. Implicit algorithms have advantages in terms of accuracy and numerical stability over explicit methods. However, the drawbacks of implicit algorithms are the additional computation requirements and then their inappropriateness for real-time applications. The other reason that the implicit methods are not suitable for real-time application is that the execution time required for the iterative solution of is unpredictable, since an input value from a future time is required in the form of \( n+1 \), which is not available at time step \( n \) when the integration step must be performed. Therefore only the explicit algorithm is referred to if there is no additional denotation.

A wide variety of other types of numerical integration algorithms are available, many of which possess unique attributes that are valuable in specific applications. The Runge-Kutta algorithm family is particularly good at simulating systems. To achieve acceptable accuracy, integration algorithms of second through fourth order are commonly used instead of the first order algorithms discussed above.

### 3.3 Hybrid Simulation Algorithm Design

#### 3.3.1 Mechanism and Representation of Event

Hybrid systems contain both continuous and discrete state variables. Within a given discrete state, the continuous variables evolve according to a set of differential (or difference) equations. Changes from one mode of operation to another, called transition or switching, are caused by state events or simply events. The transition from a location is enabled when the continuous state \( x \) or discrete state \( u \) satisfy the guard conditions, while during the transition the continuous state \( x \) jumps to a value \( x' \) given by the relation of vector fields. Suppose at the initial point \( t_0 \) or either point \( t_i \), \( t_0 < t_1 < ... t_s < ... \) such that one of the equations

\[
s_j(x(t),u(t)) = 0 \quad j = 1,2,...,m
\]  

(3.14)
is satisfied at \( t = t_s \). The \( s_j \) is called event or guard function and event \( j \) is said to occur at \( t_s \) when \( t_s(x(t_s), u(t_s)) \) is a root of the \( j^\text{th} \) event function. In the thesis it is supposed that the event functions define a serial of boundaries or surfaces, the trajectory is called to hit the surfaces when the event functions are satisfied. For a node \( q_i \) of hybrid system, if a event function \( s_j(x(t), u(t)) \geq 0 \) for all \( j \), then the integration process continues in the same node with the same differential equations \( \dot{x} \). If there exist an integer \( j \) and a time \( t' \), such that \( s_j(x(t'), u(t')) < 0 \), then the system is switched to another node \( q_k \), where the integer \( k \) that determines the new node is determined by the event function \( s_j \).

Theoretically, for a hybrid system with \( n \) nodes, there are \( n(n-1) \) possible transitions. But out of these, usually only a small portion has a real physical meaning. Therefore, it is important that the definition of the transition conditions contains as much system information as possible, because otherwise the majority of these conditions may in fact formulate only combinatorial but not physically relevant possibilities. These considerations become especially important at implementation time, where by considering only physically relevant switches the programming difficulties can be drastically reduced.

In our simulation algorithm, the event functions \( s_j \) are exclusively defined. This means that at no point \( (x(t), u(t)) \) more than one event function \( s_j \) have a sign change. In the following \( s_j(x(t), u(t)) \geq 0 \) means that for exactly one \( j \) the condition \( s_j(x(t), u(t)) \geq 0 \) holds and for all \( l \neq j \) other event functions satisfy \( s_l(x(t), u(t)) < 0 \). Then a crossing of a surface in node \( q_i \) can be characterized by a sign change, and initiates a node change to node \( q_k \). This guarantees that these sign changes lead to a unique sequence of nodes, in other words that for all points \( (x(t), u(t)) \) at most only one event function has a sign change in a certain instant. The restriction exactly satisfies the engineering applications for practical physical system.

To permit efficient computational analysis of hybrid system, a well-defined semantics is used for modeling behaviors at switching surface. Transitions between discrete states are generally modeled with a finite automaton. These switches are triggered and initiated whenever state variables satisfy the event functions, causing the trajectory transition in the state space. An examination of real-world examples and a review of other hybrid system models have led us to an identification of transition phenomena. In this section, some explicit explanations of two classes of hybrid systems are given for which our theory and algorithm are applicable.

Autonomous switching is caused by system state evolution according to the time, including the continuous and discrete states. This can’t be forecast before it become reality. It is more suitable for described by event functions and hard to solve in reality. So much attention is focused on this type of switching. The hybrid automaton is extended and made suitable for the mathematical description of our model equations.
The autonomous switching system can be defined as:

\[
\dot{x}(t) = f(x(t), u(t), q(t)) \\
q(t) = s(x(t), q^-(t))
\] (3.15)

where

\[x(t) \in \mathbb{R}^n, \quad q(t) \in \mathcal{Q} \subset \{1, 2, ..., N\}\] .

Here

\[f(x, q) : \mathbb{R}^n \rightarrow \mathbb{R}^n, q \in \mathcal{Q}\] each globally Lipschitz continuous is the continuous dynamics;

\[s : \mathbb{R}^n \times \mathcal{Q} \rightarrow \mathcal{Q}\] represents its finite dynamics.

The notation \(t^-\) may be used to indicate that the finite state is piecewise continuous from the left side of \(s(x(t), q(t))\) . Likewise \(q^- (t) = s(x(t), q(t))\) denotes it is piece wise continuous from the right side. \(q(t)\) is node mark which means the state variables exist in this node. Here \(q^- (t)\) and \(q^+ (t)\) are used to represent the vector fields of \(f(x(t), u(t))\) separated by event function \(s(x, t)\) to avoid making distinctions. The symbol \(q^- (t)\) is used to denote the predecessor of \(q(t)\) and the successor of \(q(t)\) is \(q^+ (t)\). Suppose the initial state of system is \(x^i \in [x_0, 1]\), the continuous state trajectory \(x(t)\) evolves according to \(\dot{x} = f(x, t)\) . If \(x(t)\) hit some \((s(x, t))^{-1}\) at time \(t_f\), then the state becomes \([x(t_f), f]\) , from which the process continuous. Clearly it is an instantiation of autonomous switching.

Controlled switching is the phenomenon where the location changes abruptly in response to a control command. This can be interpreted as switching between differential constrains. Controlled switching is absolutely and when it arises it allows the system to pick among a number of differential equations. It is easy to handle in reality. From the theory aspects the controlled switching can be define as:

\[
\dot{x}(t) = f(x(t), u(t), v(t))
\] (3.16)

where the definitions are same as above except that \(v(t) \in \mathbb{R}^m\) . Here \(v\) is a controlled variable input from outside to control the system trajectory evolution. A transmission control problem describes the controlled switching and continuous time system.

Likewise, discrete-time autonomous and controlled switching systems can be defined by replacing the ODE’s above with difference equations. Also, adding controls, both discrete and continuous, is straightforward.

For the controlled switching pattern, referring to such a time as a switching time, the location will transit from \(q_i\) to \(q_j (i \neq j)\) . If \(t^-\) is the begin of switching time then there is a unique end of switching time defined as \(t^+\) . The difference between \(t^-\) and \(t^+\) are refereed as switching interval. The actual transition dynamics is often assumed take place
on a much faster time scale so that the system switching interval is very short. And
furthermore, the behavior of the continuous system while it is transiting between the
different discrete states is predictive when state variables are evolving. In this case the
system spends relatively little time and resource in the transition and it is relatively easy to
handle in practical system. When the state satisfies certain conditions, described by the
state’s event function in a specified subspace of the state space, the transition will begin.
Also, when a transition occurs, there is some means of determining the initial state of the
new dynamical system. It is a direct transition from one location to another. In digital MPU
running numerical integration algorithm, it is often completed in one step integration.

The autonomous switching is more complicated in the case of location transition.
Since the transition from one location to another is autonomous, it can’t be transited by
imposed force. Therefore there are other motions around the switching surface when the
event function is satisfied. These situations are deeply relative to the event detection and
location processing and they are described in detail in the following sections.

3.3.2 Event Detection and Location

Particular interest to the simulation of hybrid system is how to design the motions of
discrete transitions. A framework that encapsulates both the important continuous and
discrete dynamics of the system and their patterns is proposed. This framework identifies
more transition phenomena of autonomous transition of hybrid systems.

At a certain time \( t \), the location \( q_i \) is active and the continuous system flows
according to the differential equation \( \dot{x}(t) = f_j(x(t), u(t)) \) with initial conditions. Once the
event function \( s(x(t), u(t)) = 0 \) is crossed, the transition from \( q_i \) to \( q_j \) is enabled; the
state may be reset and the system enters mode \( q_j \) where it flows according to
\( \dot{x}(t) = f_j(x(t), u(t)) \). The problem concerned with is correctly detecting the discrete
transitions. In cases in which the new location has to be selected partly on the basis of
information from the continuous state, dealing with a non-constant mapping from a
continuous domain to a discrete domain is necessary. Such a mapping can never be
continuous and so one will have to live with the fact that in some cases decisions will be
very sensitive.

Solving a differential equation in the analytical sense means to find a continuous
function. In the case of a numerical solution one must generate a discrete set
\( \{x_0, x_1, \ldots, x_k, \ldots\} \), which is called mesh points, of the state values which approximate \( x(t) \)
at the set of discrete times \( \{t_0, t_1, \ldots, t_k, \ldots\} \). It is usual to test the values of each event
function for different sign at \( x_i \) and \( x_{i+1} \). A change of sign in any one indicates that an
event has occurred in \( [x_i, x_{i+1}] \). But the popular Euler and Runge-Kutta formulas produce
only approximate solutions in the certain location at the mesh point \( x_i \). This approach to
the event location problem has been an important reason for the recent work aimed at
providing these formulas with polynomial approximate solutions valid for all \( [x_i, x_{i+1}] \).
The way of locating events just described is so natural that many persons may be thinking that the task is easy. This is far from true (Ruohonene, 1994). The mesh point \( x_i \) is chosen to provide efficiency approximations to event function of a specified accuracy. Because the event function don’t influence the selection of the mesh points, the approximation may not be at all appropriate for locating the positions of the events. Even if a presence of an event is noticed, there is in general difficult to be certain whether it is first happen. The Figure 3.4 and 3.5 illustrate the exactly motion around the event function and the numerical approximation around the event function.

![Figure 3.4 Exact motion on the sliding surface](image)

![Figure 3.5 Numerical approximation of crossing](image)

In addition to these more theoretical concerns, the problem is hard to process in practical numerical computation. The root of the difficulty is that there exists no satisfactory to control the global truncation error during the course of a numerical integration. Instead it is assumed that, even the method is stable, the local errors are proportional to the global errors. This means that merely adjusting internal parameters of an algorithm can have dramatic affects on the simulation results. For example, different integration algorithms may produce qualitatively different results when applied to the
same problem. It is noted that these concerns must be considered in addition to the restrictions that normally arise when performing numerical computations. Traditional approaches often reduce the step size on the switching surface. The kernel of hybrid system simulation is how to process the transition between two continuous parts, especially with numerical errors which are unavoidable.

Besides all these fundamental difficulties there is another that arises from the polynomial approximations produced by Runge-Kutta algorithm and one-order approximations by Euler. The approximations produced by these methods don’t connect the mesh points \( x_i \) to form a globally smooth function. Therefore the first derivative has jumped non-smoothly on the switching surface. Sometimes these problems can be dealt with reliably for the non-stiff systems under small integration step size, but should be avoid for stiff problems and big integration step size. The jumps seen in these situations are comparable in rate to the step size and local error tolerances. Obviously the scheme described for locating events can exhibit anomalous behavior when such approximated solutions are used. The numerical error at any step will depend on those at earlier steps, also sometimes the error will increase as the number of steps increases. Suppose that there are two formula defined in equation (3.5) on the sides of \( i-th \) switching surface controlling the evolution of the system, which can be solved numerically using the increment formula

\[
x_{n+1} = x_n + h \varphi(t_n, x_n, \dot{h})
\]  

(3.17)

On both sides yielding a sequence of values \( \{x_n \mid n = 0, 1, 2, \ldots, N \} \). \( \varphi(t_n, x_n, \dot{h}) \) is a formula of common numerical algorithm. The step size can be constant or more general variable. The true solution of the problem is taken to be \( x = x(t) \). The general error of step \( n + 1 \) can be defined as:

\[
e_{n+1} = x_{n+1} - u(t_{n+1})
\]

(3.18)

where \( u(t) \) is the local true solution satisfying \( \dot{u} = f(t, u) \) \( u(t_0) = x_0 \). This is the error associated with a single step and it is the origin of trajectory chatting on the switching surface. Numerical chatting is theoretically can be defined as: If in the integration process a repeated sequence of location changes occurs through the locations \( q_1, q_2, \ldots, q_i \) with \( i \in \{1, \ldots, M \} \) on interval of length \( L \leq L_{\text{min}} \), where \( L_{\text{min}} \) is the length of the smallest interval of set.

Suppose that the trajectory evolves according to differential equations and approaches to the switching surface from one side, then it will hit the surface after finite integration steps. When the trajectory is close to the surface at \( n-th \) step, it is possible to get the numerical solution \( x_{n+1} \) of \((n+1)-th\) step following the formula (3.17).
Because the numerical error defined by formula (3.18) exists, in some case the switching will occur. This is called illegal switching. If this situation continues the chatting begins.

As it is shown in Figure 3.6, it is possible to get the trajectory vector $f_n$ at integration time $t_n$, it should be $f_{n+1}$ at time $t_{n+1}$, but because of the switching condition is satisfied, the time derivative will jump to $f'_{n+1}$ so that the trajectory vector jumps to $f_{n+1}$ abnormally. This means that the trajectory crosses the switching surface without controlled switching or autonomous switching, just caused by numerical errors that are not desired. This situation occurs frequently when the numerical error is enlarged if the differential equations are stiff or the integration step is increased.

The event detection algorithm should discriminate the illegal transition caused by the numerical errors and do other processing. The implications of event detection are significant, since a hybrid execution can be viewed as repeated concatenations of shorter hybrid executions, the final conditions of previous executions are the initial conditions for later executions. This means that small errors in detecting and computing the precise
state at the time of early transitions can propagate in unpredictable ways, creating unbounded simulation errors in computing later states. Several survey articles concern (Mosterman, 1999; Kowaleski et al., 1998) accurate event detection as one primary for hybrid simulators. Obviously the failure to detect the occurrence of an event or the incorrect judgment of the type of event will result in the instability when simulate hybrid system.

It was shown that the proper way to simulate any hybrid system is to numerically integrate all of the differential equations until \( t \) — the first time at which \( s(u(t), x(t)) = 0 \). At this point, the numerical integration is stopped, any applicable location switches are activated and the integration may be restarted, using \( x(t) \) as the new initial condition. This technique is widely accepted as the standard hybrid system simulation methodology. This requirement of stopping the integration precisely when \( s(u(t), x(t)) = 0 \) gives rise to the event detection problem. As described in the previous section, since numerical integrations are performed in discrete time, \( t \in \{ t_1, t_2, \ldots, t_n \} \) with step sizes \( h = t_k - t_{k-1} \), it is difficult to find \( t \) exactly. Even the time of transition is detected precisely, the approximation of transition formula is also a difficult problem. Much work has been done on the problem and most reliable approaches use interpolation to approximate the state between steps and then check the interpolation to find the time of zero crossings of \( s(u, x) \) (Shampine et al., 1991; Park and Barton, 1996; Esposito et al., 2001b; Bahl and Linninger, 2001). It is well known that, when simulating hybrid systems, a failure to detect an event can have disastrous results on the global solution due to the nonsmooth nature of the problem (Branicky, 1995). Works in (Mosterman, 1999) primarily concern the accurate event detection for the requirements of hybrid simulators. Event detection in hybrid systems is, in itself, a very difficult problem (Shampine et al., 1991).

Here the main consideration is the events caused by autonomous switching, which is unpredictable before happening. Common numerical solution algorithm may not realize it when it happens and will produce an inaccurate solution. Some mathematical tools for solving differential equations try to recognize and cope with this problem. In these approaches the event is detected by finding a step size for which a step is from \( t' \) to \( t^* = t' + \sigma h \quad 0 < \sigma \leq 1 \). From the calculation of \( s(x(t^*), u(t^*)) \to 0 \) by almost infinite loops, the event can be detected. They are designed to detect the discrete events under the desirable tolerance, by rejecting the current step, and trying again with a small step size. It is often the case that such algorithms will repeatedly try steps that satisfy the event and, on their rejection, step size falls shorter but the trajectory succeeds. In this way, eventually the integration step size on the switching surface becomes so small that the result is accurate. But the obvious shortcoming of these algorithms is that they are too time cost, especially when the system equations are stiff.

A proper method of event detection and ability to avoid this inefficiency and unreliable
handling of the discontinuity is proposed. It is an important issue of managing the tradeoff between efficiency and the desired accuracy by varying the step size in the fields of numerical integration. Large step size results in more efficient simulations but decrease the accuracy and stability of the results. Small step size is required to maintain stringent accuracy requirements when the solution to the differential equation is ill-behaved, however the computations become extremely expensive due to the large number of steps needed to complete the simulation. Therefore from the practical consideration, the algorithm of high speed and desirable accuracy are developed to find an excellent tie from these two aspects.

A first issue of great practical importance is whether the solver can detect the event precisely. In our approach the algorithm monitor the sign of the event function and search for an event in the span of a step from an approximation of \( x(t) \) at \( t_k \) to the approximation of \( x(t) \) at \( t_{k+1} = t_k + h \) by fixed step size. When the event function
\[
s(x(t_k), u(t_k)) \times s(x(t_{k+1}), u(t_{k+1})) < 0
\]
i.e. the sign of event function changes from positive to negative or from negative to positive, an event happens and the system trajectories cross the switching surface. A first-in-first-out stack is used to store the calculated approximation of \( x(t) \) at \( t_k \) and \( t_{k+1} \). The integration formula is the current formula at time \( t_k \). The current \( x(t_k) \) is calculated and push into stack. This value is regarded as the initial value of \( x(t_{k+1}) \) for the next step approximation. If there is no event occurs, the next step approximation of \( x(t_{k+1}) \) will also be push into stack to construct the normal trajectory. Otherwise, this value will be pop out from the stack and the algorithm will turn to event location processing.

Here the parlance of crossing is used to distinguish the traditional approach. The proposed approach is clearly efficient than the traditional ones while it is still face a problem. Because it separates the event detection and event location, much work is left to event location algorithm.

Locating event is an important aspect in the procedure of simulating hybrid system using the event driven scheme. Event location is mainly designed to solve the problem of what happened on the switching surface, as accurately as possible with a given tolerance. Section 3.1 introduces the mathematic model of hybrid system and it correspondent engineering aspects concisely, which present the character of hybrid system from the theoretical and practical aspects. Here a detail description of types of trajectory movement is given when the trajectory is closing to the switching surface, which provides a strong background for the study of event location mechanism.

Consider the dynamic system can be written as equation (3.5). For analysis convenience assume that there only exists one switching surface dividing the state space into two regions, \( S_i \) and \( S_j \), where the vector fields belong to two regions and separated by the switching surface defined as a scalar function \( S_{ij} \) such that:
Here the influence of autonomous switching is only considered, which is only decided by the state variable $x$ and let the input $u$ to be integrated into the system and mapped by $x$. And also define the two regions:

$$S_i = \{ x \in \mathbb{R}^n | s_y(x(t)) > 0 \} \quad \text{and} \quad S_j = \{ x \in \mathbb{R}^n | s_y(x(t)) < 0 \}$$

Figure 3.7 Trajectory motion around the switching surface.

In the Figure 3.7a, this system can be looked as a hybrid system with two locations and each location has some constrained equations. The trajectory will only evolve in its own location till the event function is triggered. From the state space theory, the state space is divided into two subspace $s_i$ and $s_j$ by the switching surface $s_{ij}$ such that $\Gamma^w = s_i \cup s_{ij} \cup s_j$. The vector fields push the trajectory to $s_{ij}$ in the subspace $s_i$ (leave location $i$) and push the trajectory from $s_{ij}$ into the subspace $s_j$ (enter location $j$).

Suppose the initial state of this system is in $s_i$, the trajectory will hit surface $s_{ij}$ in some time, cross it transversally and proceed in $s_j$. This is called a transversal mode. Note that the definition of transversal refers to the solution which is transited to $s_{ij}$ and doesn’t refer to the vector fields $f$. Any solution of this system with an initial value in $s_i$, with the character of transversal, exists and is unique. In the Figure 3.7b, the trajectory will begin from one location, move to the other under the control of vector fields. When the event function is satisfied, which means the trajectory hit $s_{ij}$, the solution can’t leave it and will therefore move along the surface. This is often called a sticking mode. This behavior will cause chatting on the sliding surface since the trajectory transits back and forth around it.

In the Figure 3.7c, according to the procedure of trajectory evolution, the state variables will still stay on the switching surface under some conditions or the solution which starts close to $s_{ij}$ will move away from it and entering either $s_i$ or $s_j$. This type of vector fields around the switching surface is named repulsive mode since the trajectory is repulsed from $s_{ij}$. The solution still exists but it is not unique in forward time and it is
decided by the moving modes of vector fields.

If the vector fields $f_i$ and $f_j$ are locally both pointing away from or towards the switching surface $S_y$, the dynamics is assumed to be locally constrained to the surface, just as depicted in the Figure 3.7b and Figure 3.7c. The open subset $S^\circ_y$ of the switching surface is often referred to as the sliding surface. The sliding surface $S^\circ_y \subset S_y$ is defined as:

$$S^\circ_y = \{x \in S_y \mid -1 < \mu_y(t,x) < 1\} \quad (3.21)$$

Correspondingly the surface

$$S^\circ_c = \{x \in S_y \mid \mu_y(t,x) > 1 \cup \mu_y < -1\} \quad (3.22)$$

is defined as crossing surface.

The surfaces defining the borders of the sliding surface are thus given by

$$M^1_y = \{x \in S^\circ_y \mid \mu_y(t,x) = -1\} \quad \text{and} \quad M^2_y = \{x \in S^\circ_y \mid \mu_y(t,x) = 1\} \quad (3.23)$$

The parameter $\mu_y(x)$ is a parameter which defines the convex combination $f_y$ of vector fields on the regions of $S_i$ and $S_j$. A detail explanation will be given in the following section. The graphic explain is shown in Figure 3.8:

Figure 3.8 Definitions on the switching surface.

The surface $M^1_{ij}$ and $M^2_{ij}$ divide the state space into three regions which are defined.
as:

\[ M_{ij}^{c1} = \{ x \in \mathbb{R}^n \mid \mu_{ij}(t, x) < -1 \} \]
\[ M_{ij}^{c2} = \{ x \in \mathbb{R}^n \mid -1 < \mu_{ij}(t, x) < 1 \} \]
\[ M_{ij}^{c3} = \{ x \in \mathbb{R}^n \mid \mu_{ij}(t, x) > 1 \} \]

(3.24)

The surfaces \( M_{ij}^{c1} \) and \( M_{ij}^{c2} \) defining the borders of sliding surface can also be considered as denoting the events in hybrid system. They are the new events added in the procedure of event location. These new event detection mechanisms will be used to control the system trajectory on the switching surface. These event functions can be defined as:

\[ \mu_{ij}(t, x) = 1 \quad \text{and} \quad \mu_{ij}(t, x) = -1 \]

(3.25)

They are the new events defined in the procedure of event location. These events have the properties and actions similar to the basic event and they can be called induced events. These events can also be triggered and influence the system trajectory greatly.

Figure 3.9 Division of state space by event functions
As mentioned before, the state space $\mathbb{R}^n$ around the switching surface $S_i$ can be divided into two regions named $S_i$ and $S_j$, satisfying $\mathbb{R}^n = S_i \cup S_j$. Furthermore, the board surfaces $M^i$ and $M^j$ of sliding surfaces can also divide the state space into three regions of $M^i$, $M^j$, and $M^{ij}$. Therefore, the state space is divided to satisfying $\mathbb{R}^n = M^i \cup M^j \cup M^{ij}$. Because the state variables present same properties in the regions of $M^i$ and $M^j$, they are marked by one symbol $M^i_j$. These regions aren’t exclusive and there exist overlap among them. This means that the vector variable $x$ can belong to different regions simultaneously in a certain time with the constraint of vector fields. The entire regions and surface are shown as Figure 3.9.

This division into four regions makes it relatively straightforward to implement numerical algorithm and reduces the number of checks that have to be made every time when the event is triggered. This way of proceeding is not only more efficient, but it furnishes a reliable solution of the task it intend to. Since two surfaces $M^i$ and $M^j$ given by event functions $\mu_i(x) = 1$ and $\mu_j(x) = -1$ are added, it makes the algorithm more robust for the location of events when the trajectories hit the switching surface.

The event functions divide the entire space into several regions and in each region there exist a set of differential equations to control the evolution of trajectories. The event functions manage the transition between two regions when the transition conditions are satisfied. Since there are two types of event functions in our approach, the properties and operations of each event function are explained in detail.

If $x \in S_i$ or $x \in S_j$, there are three surfaces the trajectories have to look for, namely the switching surface $S_i$ and the sliding boundaries $M^i$ and $M^j$. Therefore, the nature choices for event functions are $s_i(t, x) = 0$, $\mu_i(t, x) = -1$ and $\mu_j(t, x) = 1$. According to the regions where the state variables belong to and which the event functions they will trigger, the state variables will evolve to different regions or surfaces using correspondent integration formula.

1. Suppose the vector variables stay in the region $S_i$ at the time $t_i$ currently. At the next time $t_i + 1$, the event function $s_j(t, x)$ changes from positive to negative and the event is triggered, which means that the vector fields will be changed to different ones. After that, the event detection and location mechanism should also check whether the event $\mu_j(t, x) = -1$ or $\mu_j(t, x) = 1$ occurs. There two possible situations exist. On one hand, the trajectories evolve from $\mu_j(t, x) < -1$ (the left side of $\mu_j(t, x) = -1$) to $\mu_j(t, x) = -1$; or the trajectories evolve from $\mu_j(t, x) > 1$ (the right side of $\mu_j(t, x) = 1$) to $\mu_j(t, x) = 1$. In both cases, the trajectories will stay in the region $|\mu_j(t, x)| < 1$ after they cross the switching surface. Therefore, the trajectory will switch to and stick on the sliding surface defined by $S_j$ and integration formula $f_j$ is used to calculate the system evolution. On the other hand, the trajectories can reach the boundary surface $\mu_j(t, x) = -1$ from the right hand or reach the boundary surface $\mu_j(t, x) = 1$ from the left hand, which means the
trajectories don’t stay the region \( |\mu_y(t,x)| < 1 \) and switch out to the region \( \mu_y(t,x) < -1 \) or \( \mu_y(t,x) > 1 \). Therefore after the trajectories hit the crossing surface they will switch to \( S_j \) directly and use \( f_j \) to calculate the system evolution. On the contrary, the event \( \mu_y(t,x) = -1 \) or \( \mu_y(t,x) = 1 \) probably occurs before the event \( s_y(t,x) = 0 \) occurs. Note that in this situation the events \( \mu_y(t,x) = -1 \) and \( \mu_y(t,x) = 1 \) are induced in the procedure of locating event \( s_y(t,x) = 0 \). Therefore these two events are regarded affiliate to the basic event \( s_y(t,x) = 0 \). Constrained by this explanation, event detection mechanism will not check whether these two events occur before confirm that the basic event \( s_y(t,x) = 0 \) has taken place. The method decreases the times of event checking and increases the efficiency of system solution.

2, Suppose the vector variables initially stay in the region \( S_j \), the system will run similarly to the first situation. After detecting an event of crossing the switching surface, it will check whether the trajectory transit to the region of \( |\mu_y(t,x)| < 1 \) or \( |\mu_y(t,x)| > 1 \) to find what directions to evolve to. If \( |\mu_y(t,x)| < 1 \) the trajectories will switch to sliding surface, using \( f_y \) to continue the integration, otherwise it will transit to \( S_i \) directly and use \( f_i \) to continue the integration.

3, Suppose the trajectories evolve from \( x \in S^y_{ij} \), i.e. \( |\mu_y(t,x)| < 1 \), there are two surfaces to look for, namely \( M^1_{ij} \) and \( M^2_{ij} \). Since the trajectories currently on the surface \( s_y(t,x) = 0 \), so they don’t need to check this event. In this case the trajectory will evolve on the opposite direction to trigger the two events \( M^1_{ij} \) and \( M^2_{ij} \). From the region \( |\mu_y(t,x)| < 1 \), one possibility is that the trajectories will evolve to the left side till hit the boundary \( \mu_y(t,x) = -1 \), then the vector fields will switch to \( S_i \) and use \( f_i \) to calculate the integration. The other possibility is that the trajectories will evolve to the right side till it hit the boundary \( \mu_y(t,x) = 1 \), then the vector fields will switch to \( S_j \) and use \( f_j \) to calculate the integration. In this situation it is not need to check the event \( s_y(t,x) = 0 \) and use the other two events \( \mu_y(t,x) = -1 \) and \( \mu_y(t,x) = 1 \) to control the transition of vector fields.

A serial of event variables are defined to denote which region or surface the state variables belong to. Also these event variables make the solver know which event to look for and which vector fields to use. Set \( sta1, sta2 \) and \( sta3 \) to represent which regions \( S_i, S_j \) and \( S_y \) the state variables exist respectively in order to keep track of the trajectories. Similarly, to keep track of the state variables in the region \( M^1_{ij} \) and \( M^2_{ij} \), introduce the event variables \( sta4 \) and \( sta5 \) respectively. Combine the symbol \( sta = (sta1,sta2,sta3,sta4,sta5) \) and using the event detection and location mechanism introduced before, the event variables can be the following values:
\[
\begin{align*}
\text{sta1} &= \begin{cases} 
1 & x \in S_i \\
-1 & x \notin S_i 
\end{cases} \\
\text{sta2} &= \begin{cases} 
1 & x \in S_j \\
-1 & x \notin S_j 
\end{cases} \\
\text{sta3} &= \begin{cases} 
1 & x \in S_{ij}^r \\
-1 & x \notin S_{ij}^r 
\end{cases} \\
\text{sta4} &= \begin{cases} 
1 & x \in M_{ij}^c \\
-1 & x \notin M_{ij}^c 
\end{cases} \\
\text{sta5} &= \begin{cases} 
1 & x \in M_{ij}^v \\
-1 & x \notin M_{ij}^v 
\end{cases}
\end{align*}
\]

The value of events will be variable according to the trajectory evolution. When a certain event occurs, the event variable will change according to it, demoting that the state variables switch to the corresponding region or surface. For instance, assume in a certain time the state variables \( x \) belong to \( S_i \) and \( M_{ij}^c \). The trajectories will move to the switching surface and hit it after some time. Before the trajectory cross the switching surface, its state variable is as \( \text{sta} = (-1,1,-1,1) \). Suppose the trajectories reach the switching surface at a certain time and sliding on it, the state variable as \( \text{sta} = (-1,-1,1,1) \) can be achieved.

Together with the law of event variables defined in 3.26 the Table 3.1 lists which region the state variables stay in and which surface they should look for according to the evolution of trajectories. The direction from which the trajectories should evolve to the surface is also reported. The symbol → denotes the evolving to the event surface from negative direction and the symbol ← denotes the evolving to the event surface from positive direction. The symbol × denotes there is no event should be detected since this event can’t happen in this situation.

### Table 3.1 Explanation of trajectory transition

<table>
<thead>
<tr>
<th>x</th>
<th>(sta1, sta2, sta3, sta4, sta5)</th>
<th>( s_i (t, x) = 0 )</th>
<th>( \mu_i (t, x) = -1 )</th>
<th>( \mu_i (t, x) = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_i \cup M_{ij}^c )</td>
<td>(1, -1, -1, 1, -1)</td>
<td>←</td>
<td>→</td>
<td>←</td>
</tr>
<tr>
<td>( S_i \cup M_{ij}^r )</td>
<td>(1, -1, -1, 1, 1)</td>
<td>←</td>
<td>←</td>
<td>→</td>
</tr>
<tr>
<td>( S_j \cup M_{ij}^c )</td>
<td>(-1, 1, -1, 1, -1)</td>
<td>→</td>
<td>→</td>
<td>←</td>
</tr>
<tr>
<td>( S_j \cup M_{ij}^r )</td>
<td>(-1, 1, -1, 1, 1)</td>
<td>→</td>
<td>←</td>
<td>→</td>
</tr>
<tr>
<td>( S_{ij} )</td>
<td>(-1, 1, 1, 1, 1)</td>
<td>×</td>
<td>←</td>
<td>→</td>
</tr>
</tbody>
</table>

Events of controlled switching can be known exactly beforehand, then it is known the vector fields transition before the trajectories actually reach the switching point. This type of events can be triggered by time period or input commands from outside. Often the state variables are extended by adding these parameters as new variables.
In the first case the state transition time is simply registered as a predictable breakpoint. Typically, if a system is periodic, for instance due to a periodic forcing with period \( T \), it might be of interest to locate the end of the period where the time could be reset to zero. Otherwise assume \( x_u \) is the state variable of the scaled time and \( t_p \) is the time that is of interest. Introduce a surface \( \beta_p \) defined by the function \( s_p(x) = 0 \), where \( s_p(x) = x_u - t_p \). Then it is only necessary to look for \( \beta_p \) for increasing values of \( s_p(x) \). When the time \( t_p \) expired, which means the event function \( s_p(x) = 0 \) is satisfied and the time event occurs.

For the transition caused by the input commands, the transition conditions are associated with a table-like allocation function \( s(i) \), which for the location \( q_i \) respectively. The transition commands are triggered by transition condition and they will assign current location to the successor one \( q_k \) with \( i \neq k \). A lookup table or even small scale database can be used to define the relationship among the transition conditions, transition commands and correspondent source and target transition locations. Before each integration step, say from \( t \) to \( t + h \), here \( h \) is defined as current integration step size, the table will be examined. If there is an event at \( t + \delta \), where \( \delta < h \), the event will be detected at the current step and the solver will process the event according to the look-up table, i.e. transit to another location or remain in the current one.

### 3.3.3 Integration Formula on the Sliding Surface

The last section analyzes the events around the switching surface and gives the method of event location and processing. Section 3.2 gives a simple explanation of one dimensional differential equation. It is now need to define an integration formula on the sliding surface in a general sense for \( n \) dimension differential equations. Suppose that there is only one single switching surface \( S_{ij} \) dividing the state space into two regions, \( S_i \) and \( S_j \), satisfy \( \mathbb{R}^n = S_i \cup S_j \). And the vector fields \( f_i(t,x(t)) \) and \( f_j(t,x(t)) \) belong to two regions and separated by the switching surface which is defined as a scalar function \( s_{ij} = \{ x \in \mathbb{R}^n \mid s_{ij}(x(t)) = 0 \} \). The state variable \( x(t) \) is on the surface while \( s_{ij}(x(t)) = 0 \).

The function \( f_i(t,x(t)) \) and \( f_j(t,x(t)) \) is assumed to be satisfy the constrains introduced in the section 3.2 for all \( x \notin S_{ij} \). From this assumption it follows that the solution within each region \( S_i \) and \( S_j \) exists and is unique.

The set-valued extension of \( f_i(t,x(t)) \) and \( f_j(t,x(t)) \) for \( x \in S_i \subset S_{ij} \) is given by the closed convex hull of all the limits, which means the trajectories are moving on the sliding surface.

\[
F(t,x(t)) = co \{ y \in \mathbb{R}^n \mid y = \lim_{\tau \rightarrow t} f(t,\tau x(t)), x \in \mathbb{R}^n \}
\]  \hspace{1cm} (3.27)

where \( co \{ \} \) denote the smallest closed convex set containing \( \{ \} \). All the limits exist
because \( f_i(t,x(t)) \) and \( f_j(t,x(t)) \) are assumed to be locally continuous, smooth and linearly bounded for all \( x \not\in S_y \).

The system described by equation (3.5) does not define \( f(t,x(t)) \) if \( x(t) \) is on the surface \( S_y \). This problem can be overcomed with the following set-valued extension \( F(t,x(t)) \):

\[
\dot{x}(t) \in F(t,x(t)) = \begin{cases} 
 f_i(t,x(t)) & x \in S_i \\
 \text{co} \{ f_i(t,x(t)), f_j(t,x(t)) \} & x \in S_{ij} \\
 f_j(t,x(t)) & x \in S_j 
\end{cases}
\]

(3.28)

where the convex set with two right-hand sides \( f_i(t,x(t)) \) and \( f_j(t,x(t)) \) can be cast in

\[
f_y(t,x(t)) = \text{co} \{ f_i, f_j \} = \{ \eta f_i + (1 - \eta) f_j, \forall \eta \in (0,1) \}
\]

(3.29)

The parameter \( \eta \) is a parameter which defines the convex combination and has no physical meaning. The extension of a discontinuous system into a differential inclusion 3.29 is known as Filippov’s convex method (Filippov, 1964, 1988). The set-valued extension \( F \) of \( f \) should be suitable. Since the system equations are mathematical model of a physical system, then the solution concept that guarantees existence of solutions is needed.

When the state variable is on the sliding surface, namely \( x \in S_{ij} \subset S_y \), it is obviously that both trajectories of \( f_i(t,x(t)) \) and \( f_j(t,x(t)) \) have to tangent to \( S_y \) defined by \( s_y(x(t)) = 0 \). The straightforward geometric consideration is illustrated in Figure 3.10.

![Figure 3.10](image)

Figure 3.10 Graphic demonstration of Filippov construction.

In this case

\[
\langle h_y, f_y \rangle = 0
\]

(3.30)
where $h_{ij}(x)$ means the gradient of $S_{ij}^x$.

Suppose $x(t)$ is a vector with dimension $n$, then

\[
h_{ij}(x) = \left( \frac{\partial s_{ij}}{\partial x_1}, \frac{\partial s_{ij}}{\partial x_2}, \ldots, \frac{\partial s_{ij}}{\partial x_n} \right)
\]

(3.31)

and $\langle \cdot \rangle$ defines the scalar production of two vectors.

Define

\[
\eta = \frac{1 - \mu_{ij}}{2}
\]

(3.32)

then $\mu_{ij} \in (-1,1)$.

Substituting (3.32) into the equation (3.29), it can get

\[
f_{ij} = \frac{1 - \mu_{ij}}{2} f_i + (1 - \frac{1 - \mu_{ij}}{2}) f_j
\]

(3.33)

Rewrite is as

\[
f_{ij} = \frac{f_i + f_j}{2} + \mu_{ij} \frac{f_j - f_i}{2}, \quad \mu_{ij} \in (-1,1)
\]

(3.34)

Substituting this equation to equation (3.30) it is written as

\[
\left\langle h_{ij}, \frac{f_i + f_j}{2} + \mu_{ij} \frac{f_j - f_i}{2} \right\rangle = 0
\]

(3.35)

Then the solution is

\[
\mu_{ij}(t, x(t)) = \frac{\langle h_{ij}, f_i \rangle + \langle h_{ij}, f_j \rangle}{\langle h_{ij}, f_i \rangle - \langle h_{ij}, f_j \rangle}
\]

(3.36)

A simple explanation from other aspects can also be given. The state velocity vectors $f_i$ and $f_j$ are assumed to be constant for some point $x$ on the surface $s_{ij}(t, x) = 0$ within a short time interval $(t, t + \Delta t)$. Let the time interval $\Delta t$ consist of two sets of intervals $\Delta t_i$ and $\Delta t_j$, such that $\Delta t = \Delta t_i + \Delta t_j$, $f_i$ for the time from the set $\Delta t_i$ and $f_j$ for the time from the set $\Delta t_j$. Then the increment of the state vector after time interval $\Delta t$ is found as $\Delta x = f_i \Delta t_i + f_j \Delta t_j$, and average state velocity as $\dot{x}_{av} = \Delta x / \Delta t = uf_i + (1 - u)f_j$ where $u = \Delta t_i / \Delta t$ is the relative time for control to take value.
\( f_i \) and \((1-u)\) is the relative time to take value \( f_j \), \( 0 \leq u \leq 1 \). To get the vector \( \dot{x} \), the time \( \Delta t \) should be tended to zero. Then the equation \( \dot{x} = uf_i + (1-u)f_j \) can be used to represent the motion during sliding modes. Since the state trajectory during sliding mode are on the surface \( s_j(t,x) = 0 \), the parameter \( u \) should be selected such that the state velocity vector is in the tangential plane to this surface, or

\[
\dot{s} = \text{grad} \ (h(t,x)) \times \dot{x} = 0
\]

with \( \text{grad} \ (h(t,x)) = ( \frac{\partial h}{\partial x_1}, \ldots, \frac{\partial h}{\partial x_n} ) \). The solution is given by

\[
u = \frac{\text{grad} \ (h) \times f_j}{\text{grad} \ (h) \times (f_j - f_i)} \] (3.38)

The equation (3.37) is similar to equation (3.30) essentially, by submitting \( u \) with \( u = (1 - \mu_i) / 2 \), equation (3.38) is constant to equation (3.36). This can be regarded as verification of calculation for two region switching system.

The integration formula governing the movements of trajectories on the sliding surface \( S_j \) is constructed by the convex hull of \( f_i(t,x(t)) \) and \( f_j(t,x(t)) \). Furthermore the more detail formula is described by the equation (3.34). When the state variables switch to the sliding mode, these formulas are used to calculate the integration. The region \( \mu_i \in (-1,1) \) can be regarded as the boundary of sliding surface. When the trajectories exist on this region, they are defined as evolving on the sliding mode, otherwise they are on the crossing mode. This boundary is consistent what are defined in section 3.3.2 of event detection and it can used to detect whether a certain event happens.

### 3.3.4 Hybrid Automaton Expansion and Flow

As discussed in the previous sections, hybrid system simulation should be capable of generating events, managing event transition and monitoring invariants of continuous dynamics. This capability enables the interaction of discrete and continuous dynamics and makes the correct simulation of hybrid systems possible.

When simulating a hybrid system, the interaction of discrete and continuous dynamics goes through the following steps and the flow is explained as Figure 3.11.
1. During the continuous evolution in a certain location, the system is simulated as a continuous time system where the discrete states are kept constant by the continuous dynamics of guidance conditions. The events of autonomous switching and controlled switching during the simulation are monitored such that the time when the event functions are satisfied, there are no events missed.

2. At each discrete time point where the events are triggered and the behavior of the system is found, the transitions starting from the current state are evaluated.

3. If a transition is enabled, the hybrid automaton makes a state transition. The continuous dynamic of the destination state is initialized by the reset relation on the transition. The simulation continues from the new location as continuous system with the current time point.

For a deterministic hybrid automaton, the above simulation scheme will calculate the execution for each initial condition within the precision of continuous-time simulation. The execution also depends on many other factors, which include the user’s choice of integration methods, the maximum and minimum step sizes, and the error tolerance.

The first step for hybrid system simulation is to decide the numerical method of differential equations in the locations. It will be assumed that a suitable time stepping method is chosen to integrate smooth ODE systems with a desired tolerance. There are numerous methods to solve different types of ODEs, see, e.g. (Hairer et al., 1993, 1996) and based on these numerical solvers, e.g. (Brown, 1989), the first task is to choose one that is suitable for the current system. One of the most frequently used time-stepping
solving methods for differential equations is the fourth-order Runge-Kutta that works very well for non-stiff problems, but for more stiff systems a special solver is required, based on, e.g., a BDF (Brown, 1989) or Radau (Swart, 1997; Hairer and Wanner, 1999) method. For linear systems, there also exist special methods (Celledoni, 1998; Enright, 1979; Pavlov and Rodionova, 1994), which are usually faster than general solvers for nonlinear systems.

The solver inside MATLAB is used such as ODE45 and ODE113. ODE45 is based on an explicit Runge-Kutta formula, the Dormand-Prince pair. It is a one-step variable-step solver and in general is the best solver to apply as a first try for most problems. ODE113 is a variable-order Adams-Bashforth-Moulton PECE solver. It can be more efficient than ODE45 at stringent tolerances when the differential equations are stiff. Since our strategy is to develop a fast hybrid system simulator, it is need to control and monitor the execution time of the program exactly. But the inside numerical solvers can’t satisfy our requirements. After simulating the hybrid system using the inside solver, explicit Euler method is used to reconstruct the simulator, in which the execution time can be controlled precisely.

The discrete transition is controlled by the event detection and location mechanism. The hybrid automaton may reside in the current location as long as the continuous state remains inside its invariant. The integration is continued till one of the transition events gives a zero crossing of event surface. Each discrete transition may be taken when the guard conditions are satisfied and the vector fields are transited to another location. The previous sections explain the event detection and location algorithm clearly. This section extends it into hybrid automaton to construct the hybrid system simulator.

Theoretically the state space $\mathbb{R}^n$ around the switching surface is divided by three surface namely $S_i$, $M_i^j$, and $M_i^k$. These surfaces construct four regions and one sliding surface, named $S_i$, $S_j$, $M_i^j$, $M_i^k$ and $S_i^j$. Five event variables are defined to denote which region or surface the current state variables exist. In the theoretical fields, these definitions describe the character and behavior precisely. But it is not so efficient in the procedure of program development. Consider the section of event detection and section defining the integration formula again, it is found that the vector fields only transit in the differential inclusion including three differential equations. Furthermore, from the deeper analysis, there exist induced events of $\mu_i(t, x) = -1$ and $\mu_i(t, x) = 1$ in the procedure of detecting basic event of switching surface crossing. The induced events are affiliate to the basic event, which means these events will not trigger the location transition if the basic event is not triggered. They only denote the change of regions in the procedure of trajectory evolution. For example, suppose initially the state variables exist in the region $S_i \cup M_i^j$, and the event variable $sta = (1, -1, -1, -1)$ occurs, the event variables will become $sta = (1, -1, -1, -1, 1)$ and the trajectory move to region $S_i \cup M_i^k$, but it will not cause the transition of vector fields.
conclusions can be got from the deep study of event transition mechanisms.

This means that it is enough to use three event variables to denote the transition of locations. Define \( \text{state} = (\text{state} 1, \text{state} 2, \text{state} 3) \), denoting the corresponding locations in hybrid automaton. Each location contains one of the vector fields in the differential inclusion. Therefore the hybrid automaton is extended with adding a location in the basic system and in such way that each behavior including switching as a location in this automaton can be modeled, at the same time the potentially discontinuity and oscillation are removed. In this case the automaton can be rebuilt from two locations to three locations around a certain switching surface. The general idea behind this construction can be seen in Figure 3.12.

![Figure 3.12 Extension of hybrid automaton in the simulator](image)

The flow chart of the algorithm is shown as Figure 3.13 with the explanation with it.

1. After each time step of the numerical integration in node \( q_i \), check whether discontinuities or crossing of surface occurred, by testing of the transition conditions \( s_{ij}(t, x) = 0 \) for sign changes.
2. If no sign change has occurred, then continue the integration normally. Otherwise localize the switch point exactly via the event detection mechanism of the solver.
3. If the switching is caused by controlled transaction, such as time expiration or outside command input, switch to new node absolutely, calculate the trajectory on the new node as continuous integration with the initial value transferred from the former node.
Figure 3.13 Flowchart of hybrid simulation algorithm
4. If it is an autonomous switching, decide the situation of switching or sliding. If it is switching mode, switch to another node directly, the operation is same as in step 3, the evolving formula also switching directly from $f_i$ to $f_j$; if it is sliding mode, calculate the trajectory using formula (3.34). The detail discussion is shown below.

5. Restart the integration again and begin a new loop if not end.

When the trajectory is close to the switching surface, according to the current value it can be decided which node the current state is belonging to and the state value which represents the node of current state can be evaluated. Suppose that on a certain time the initial trajectory vector $x$ exists in the node $s_i$ and evolves to the switching surface. At the current loop the evolving formula of this node is used to calculate the state variables according the state value, till the trajectory vector hits the surface. According to the condition and calculation formula on the switching surface, if $-1 \leq u_{ijy} \leq 1$, the trajectory vector reaches sliding model $s_{ijS}$ and will remain on the switching surface until $u_{ijy} > 1$ or $u_{ijy} < -1$. In this case the evolution of trajectory is guided by the formula in the new additional node until transition condition is satisfied. If on certain integration step $u_{ijy} > 1$, the trajectory will switch to the node $s_j$ and we use $f_j$ to calculate the state variables; if $u_{ijy} < -1$ the trajectory will switch to the node $s_i$ and $f_i$ is used to calculate the state vector.

3.4 Modeling and Simulation of Intake System

3.4.1 Basics of Turbocharged Internal Combustion Engine

A heat engine converts thermal energy to mechanical work, by letting a work fluid go through a thermodynamic cycle. This involves coordinated heat transfer and volume change of the work fluid. The simplest type of heat engine accepts heat from a high temperature reservoir, outputs mechanical work, and expels waste heat to a low-temperature reservoir. A combustion engine is a heat engine in which the thermal energy is produced by combustion. Here, the chemical energy of the fuel is converted to thermal energy which powers the heat engine. An internal combustion engine is a combustion engine in which the combusting mixture of fuel and oxidant (usually air) acts as the work fluid. This necessarily involves a flow of fresh fuel-oxidant mixture into the engine, and a flow of exhaust out of the engine. A reciprocating internal combustion engine utilizes a cylinder-piston-crank arrangement in order to, for each cycle, draw a fresh batch of fuel-oxidant.

Here only consider the general four-stroke internal combustion engine in automotives, as it is shown in Figure 3.14. The most common internal combustion engine is gasoline powered. Others include those fueled by diesel, hydrogen, methane, propane, etc.
Engines typically can only run on one type of fuel and require adaptations to adjust the air/fuel ratio or mix to use other fuels. In a gasoline engine, a mixture of gasoline and air is sprayed into a cylinder. This is compressed by a piston and at optimal point in the compression stroke, a spark plug creates an electrical spark that ignites the fuel. The combustion of the fuel results in a lot of heat and much high pressure in the limited cylinder space, then the boosted gas drives the piston back down. This combustion gas is vented and the fuel-air mixture is reintroduced to run a second cycle. The outward linear motion of the piston is ordinarily harnessed by a crankshaft to produce circular motion. Valves control the intake of air-fuel mixture and allow exhaust gasses to exit at the appropriate times.

A turbocharger increases the pressure of the air in the inlet manifold of the engine. As the air pressure is greater than atmospheric pressure it has a higher density i.e. more oxygen per liter of air. A greater mass of air rushes into the cylinder when the intake valve in the cylinder opens. The more air means that more fuel can be added. Therefore, more power can be gotten from each explosion in each cylinder. A turbocharged engine produces more power overall than the same engine without the charging. This can significantly improve the power-to-weight ratio for the engine. At high altitudes the turbocharger rotates faster to increase delivery of air to the engine then in some case a turbocharged engine produces clean emissions.

Figure 3.14 Structure of turbocharged internal combustion engine
In simple terms, a turbocharger comprised of a turbine and a compressor connected by a common shaft supported on a bearing system. The turbocharger converts waste energy from the exhaust manifold into compressed air which it pushes into the engine. This allows the engine to produce more power and torque and improves the overall efficiency of the combustion process. The turbocharger is bolted to the exhaust manifold of the engine. The exhaust from the cylinders spins the turbine, which works like a gas turbine engine. The turbine is connected by a shaft to the compressor, which is located between the air filter and the intake manifold. The compressor pressurizes the air going into the pistons. The exhaust gas is guided into the turbine wheel by the turbine house and drives the turbine. Once the gas has passed through the blades of the wheel it leaves the turbine house via the exhaust outlet area. The speed of the engine determines how fast the turbine wheel spins. The more exhaust that goes through the blades, the faster they spin. If the engine is in idle mode the wheel will be spinning at a minimal speed. On the other end of the shaft of the turbine is attached to compressor wheel by a forged steel shaft. The compressor pumps air into the cylinders. The compressor is a type of centrifugal pump -- it draws air in at the center of its blades and flings it outward as it spins. The structure of intake system with turbocharger is shown as Figure 3.15.

![Figure 3.15 Structure of intake system with turbocharger](image_url)

The turbocharged engine can provide more power output and has better emission situations, but it is difficult to analyze and design in practical. Here some additional parts which are not equipped on the conventional engine are introduced. These equipments will be reflected in the model of intake system and have much influence on the design of simulation method.
With air being pumped into the cylinders under pressure by the turbocharger, and then being further compressed by the piston, there is more danger of knock. Knocking happens because as you compress air, the temperature of the air increases. The temperature may increase enough to ignite the fuel before the spark plug fires. Cars with turbochargers often need to lower the temperature of intake air and avoid knock partly. The other duty of intercooler is to increase the air mass flowing into the cylinder. An intercooler is an additional component that looks something like a radiator. The intake air passes through sealed passageways inside the cooler, while cooler air from outside is blown across fins by the engine cooling fan. When air is compressed, it heats up; and when air heats up, it expands. So some of the pressure increase from a turbocharger is due to heat the air before it goes into the engine. In order to increase the power of the engine, the goal is to get more air molecules into the cylinder, not necessarily more air pressure.

Most automotive turbochargers have a wastegate, which allows the use of a smaller turbocharger to reduce lag while preventing it from spinning too quickly at high engine speeds. The wastegate is a valve that allows the exhaust to bypass the turbine blades. The wastegate senses the boost pressure. If the pressure gets too high, it could be an indicator that the turbine is spinning too quickly, so the wastegate bypasses some of the exhaust around the turbine blades, allowing the blades to slow down.

### 3.4.2 Intake System Model of Turbocharged Engine

Knowledge of port air-mass flow is important for accurate air/fuel ratio control on spark ignite engines. Unfortunately it is not directly measurable and therefore it is estimated from the parameters in the intake manifold (Kotwick et al., 1999; Jankovic and Magner, 1999). Given the engine speed and the pressure in the intake manifold, the air-mass flow to the cylinder can be estimated. But there is considerable pressure dynamics in the intake system describing by a model. Intake manifold pressure estimation is a well-studied topic for naturally aspirated engines. On turbocharged engines there is an additional challenge as the turbocharger couples the intake side and exhaust side when the wastegate is operated. It is desirable to open the wastegate at part load to reduce the pumping losses and thereby improve fuel economy. Changes in wastegate opening will act as a disturbance that influences the estimated intake manifold pressure. Thus, this section focuses on modeling of intake manifold pressure and no consideration for air charge efficiency.

On the modern engine the air-mass flow sensor is located after the air filter and there are several large volumes between the intake manifold and the sensor. Each volume introduces filling and emptying dynamics, which distort the measurement. One method to deal with these volumes is to model the entire intake system between the sensor and the intake manifold. On naturally aspirated engines, this approach has been applied in
(Grizzle et al., 1994). Unfortunately, this is harder to implement on turbocharged engines as several of the components are complex such as the compressor and the intercooler.

When modeling a turbocharged spark ignite engine intake system by using physical modeling, it is beneficial to divide the engine in distinct subsystems. The mass flow through the engine is central in the modeling and therefore the path of the flow will be described here. The air enters the engine via the air filter, where dust and other particles are removed. The clean air then enters the compressor where both pressure and temperature increase. The air temperature must be lowered in order to avoid knock. This is done with a heat exchanger, called intercooler. The mass flow into the cylinders and thereby also the output power of the engine is controlled by the throttle. The throttle opening depends on the accelerator pedal position and the mass flow through it is modeled. After the throttle, the air enters the intake manifold where the fuel is injected. Then the air-fuel-mixture goes into the cylinders where the combustion takes place. The air flow structure of intake system through compressor, throttle, intercooler and manifold is shown in Figure 3.16.

As the compressor wheel spins, air is drawn in and is compressed as the blades spin at a high velocity. The compressor house converts the high velocity, low pressure air stream into a high pressure, low velocity air stream. The compressor mass flow, pressure ratio and turbine shaft speed vary a lot between different loads. Compressor model can be constructed by curve fitting using the data provided by the turbocharger manufacturer or achieved from the engine test bed. Those used in this study are two dimensional look-up tables and the data from experiment and experience.
\[ \dot{M}_{cm1} = map(P_{in} / P_a) \] (3.39)

where

\( M_{cm} \) is the air mass flow into the intercooler, only the span is illustrated;
\( P_{in} \) is the pressure around the intercooler;
\( P_a \) is the atmosphere pressure.

Air entering the compressor at a temperature equals to atmosphere, however it leaves the compressor cover at a temperature up to 200 degrees Celsius. Because the density of the air decreases as it is heated up, even more air can be forced into the engine if the air is cooled after the compressor. This is achieved by cooling the charge air either with water or air using intercooler.

In the intercooler the air flows through a set of thin tubes. These tubes can be regarded as a static flow restriction. The dynamic properties of the intercooler are mainly due to the thicker tubes which connect the intercooler with the compressor and the throttle. A straightforward way to model dynamics of this type is to use the method of emptying and filling tanks. The basic idea is to study mass conservation where the difference between the flow into the tank and the flow out of the tank builds up or reduce the pressure. This can be calculated with the ideal gas law:

\[
\frac{dp}{dt} = \frac{\dot{M}_{cm} - \dot{m}}{V_o} \]

(3.40)

\[
\frac{dP}{dt} = \frac{kRT}{V_o} \dot{M}_{cm} \frac{kP_a}{V_o \rho_o} \dot{m}
\]

(3.41)

where

\( \rho_{co} \) is the air density around the intercooler;
\( \dot{m} \) is the air flow rate into the throttle;
\( T_a \) is the air temperature;
\( k \) and \( R \) are the air constant.
In gasoline engine a throttle is used to control the air mass flow into the cylinders. The mass flow through the throttle can be modeled like the flow of ideal gas through a ventricle.

\[ \dot{m}_t = C_t(A_t(\theta))\phi(P_m, P_w) \]  

(3.42)

where

- \( C_t(A_t(\theta)) \) is the discharge coefficient which depends on the shape of the flow area \( A_t(\theta) \), only the span is illustrated;
- \( \theta \) is the throttle plate angle;
- \( P_w \) is the pressure around the manifold.

\[ P_{in} \geq P_m \]

\[ \phi(P_{in}, P_m) = \sqrt{P_{in}P_m} \left[ \frac{k+1}{2k} \left( \frac{k}{k+1} \right)^2 - \frac{P_m}{P_{in}} \frac{1}{k+1} \right] \]

\[ P_{in} < P_m \]

\[ \phi(P_{in}, P_m) = -\sqrt{P_{in}P_m} \left[ \frac{k+1}{2k} \left( \frac{k}{k+1} \right)^2 - \frac{P_m}{P_{in}} \frac{1}{k+1} \right] \]  

(3.43)

where

- \( \rho_m \) is the air density around the manifold;

The dynamic properties of the intake manifold can be modeled with the same technique that was used for the dynamics of the intercooler.

\[ \frac{d\rho_m}{dt} = \frac{(\dot{m}_t - \dot{M}_{cm2})}{V_m} \]  

(3.44)
\[
\frac{dP_m}{dt} = \frac{kRT_a}{V_m} \dot{m}_a - \frac{kP_m}{V_m P_m} M_{cm2}
\]  

(3.45)

Where

- \(V_{in}\) is the volume of the intercooler;
- \(V_m\) is the volume of the manifold;
- \(T_a\) is the air temperature;
- \(M_{cm2}\) is the air mass flow into cylinder.

The air flow into the cylinder can be obtained from the cylinder model:

\[
M_{cm2} = Q(P_m, P_c) 
\]

(3.46)

where \(P_c\) is cylinder pressure.

### 3.4.3 Simulation of Intake System Model

The proposed algorithm is used to solve the intake manifold model of turbo charged engine, with the requirements of high stability and less calculation time for real time control. Common numerical analysis is hampered by the steep gradient near the discontinuities. Traditional integration schemes such as Runge-Kutta method are very sensitive to the steep gradients that occur at these discontinuities and may perform poorly. In this model, \(x\) is 4-dimensional state space of \(P_{in}\), \(P_m\), \(\rho_{in}\) and \(\rho_m\), and the throttle angle is variable. The model equations are discontinuous (non-smooth) on the right hand side. In this case the trajectory vector will orient towards the switching surface \(s(t, x(t)) = P_{in} - P_m = 0\) in the region \(S_i, S_j\) and must hit the switching surface in finite time, no matter how small the integration step is. With the varying of the throttle angles and the evolution of numerical calculation, the operation mode will switch among the three states in the mathematical system, even only two states in the physical system.

The random throttle angle inputs are shown in Figure 3.17, the time span is limited to one second for simulation convenience. The trajectories of air pressure around the intercooler and manifold are shown as \(P_{in}\) and \(P_m\) in Figure 3.18a using common Euler method with small step size. The time consumption of this method is shown in Figure 3.18b (the simulation environment is Dell 4600C, Pentium4 2.6G, 512MRAM with Windows Xp professional and Matlab 6.5). It is demonstrated that this method is not efficient enough of step size equal to 0.0001 when it is implemented in the hardware system for real time control because of large time consumption. Furthermore the stability of trajectory of model must be kept, otherwise it have to compromise the output value in the control system. The corresponding results of step size equal to 0.001 are in Figure
3.19a, Figure 3.19b, which demonstrate that the time consumption is smaller but the trajectories are not stable, it is called as chatting or oscillation.

![Graph showing random throttle angle input](image)

Figure 3.17 Random throttle angle input
Figure 3.18a Trajectory of engine intake manifold pressure using common Euler method of step size equal to 0.0001

Figure 3.18b Time consumption of common Euler method on Figure 3.18a.
Figure 3.19a Trajectory of engine intake manifold pressure using Euler method of step size equal to 0.001

Number of files called: 16

<table>
<thead>
<tr>
<th>Filename</th>
<th>File Type</th>
<th>Calls</th>
<th>Total Time</th>
<th>Time Plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td>M-function</td>
<td>1</td>
<td>0.453 s</td>
<td></td>
</tr>
<tr>
<td>fixstep</td>
<td>M-function</td>
<td>1</td>
<td>0.281 s</td>
<td></td>
</tr>
<tr>
<td>vectorfields</td>
<td>M-function</td>
<td>1001</td>
<td>0.386 s</td>
<td></td>
</tr>
<tr>
<td>vectorfields/mag</td>
<td>M-subfunction</td>
<td>3003</td>
<td>0.077 s</td>
<td></td>
</tr>
<tr>
<td>hold</td>
<td>M-function</td>
<td>2</td>
<td>0.016 s</td>
<td></td>
</tr>
<tr>
<td>profile</td>
<td>M-function</td>
<td>1</td>
<td>0 s</td>
<td></td>
</tr>
<tr>
<td>revplot/ClearAllAxesNextPlot</td>
<td>M-subfunction</td>
<td>2</td>
<td>0 s</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.19b Time consumption of common Euler method on Figure 3.19a
The former solution algorithm is the Euler method of fixed step size. Here the general Euler method of fixed step size is integrated inside the proposed algorithm to calculate the continuous parts. In these two situations the integration step size can be controlled absolutely therefore the integration time can be known clearly. It is demonstrated that the calculation time is greatly reduced and the error is limited to an acceptable level. The simulation results with a very small step size are consistent to the experiment data results on the engine test bench, therefore the two simulation results can be compared directly. The algorithm is will be embedded into the real time control system, therefore the program embedded in RAM should run faster, more stable with the desirable limitation of hardware cost. The simulation results shown in Figure 3.20 to Figure 3.22 are the trajectories and relative calculation time of this model with typical throttle angle inputs.
Figure 3.20a Simulation result and time consumption using Common Euler algorithm.

Simulation time=0.1s, step size=0.0001, throttle angle =84
Figure 3.20b, Simulation result and time consumption using proposed method with inside Common Euler algorithm

Simulation time=0.1s, step size=0.001, throttle angle =84
Figure 3.21a Simulation result and time consumption using Common Euler algorithm.

Simulation time=0.1s, step size=0.0001, throttle angle =54
Figure 3.21b Simulation result and time consumption using proposed method with inside Common Euler algorithm

Simulation time=0.1s, step size=0.001, throttle angle =54

<table>
<thead>
<tr>
<th>Filename</th>
<th>File Type</th>
<th>Calls</th>
<th>Total Time</th>
<th>Time Plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td>M-function</td>
<td>1</td>
<td>0.110 s</td>
<td></td>
</tr>
<tr>
<td>fixstep</td>
<td>M-function</td>
<td>1</td>
<td>0.078 s</td>
<td></td>
</tr>
<tr>
<td>vectorfields</td>
<td>M-function</td>
<td>199</td>
<td>0.032 s</td>
<td></td>
</tr>
<tr>
<td>hold</td>
<td>M-function</td>
<td>2</td>
<td>0.016 s</td>
<td></td>
</tr>
<tr>
<td>profile</td>
<td>M-function</td>
<td>1</td>
<td>0 s</td>
<td></td>
</tr>
<tr>
<td>newplot/ObsAxesNextPlot</td>
<td>M-subfunction</td>
<td>2</td>
<td>0 s</td>
<td></td>
</tr>
</tbody>
</table>
Figure 3.22a Simulation result and time consumption using Common Euler algorithm.
Simulation time=0.1s, step size=0.0001, throttle angle =24
Figure 3.22b, Simulation result and time consumption using proposed method with inside Common Euler algorithm

Simulation time=0.1s, step size=0.001, throttle angle =24
These figures show the comparison results between the proposed method and the common numerical method. The first line of routine \textit{run} is the main routine; the executing time of \textit{run} is the time consumption of the entire simulation program. The routines below \textit{run} are the invoking routines called by \textit{run}, the running time and calling time are also listed. From theoretical analysis if the integration step size is enlarged ten times, the total calculation time should be reduced 10 times if there were no other operations to consume the CPU time using the algorithm of fixed step size. In the proposed algorithm, the total time consumption is the sum of time which includes common numerical method and the switching consumption. At last when the integration step size is enlarged ten times, the total calculation time can be reduced about eight times on Visual C++ platform. The figures describe the time consumption at typical throttle angles, which show that the calculation time is reduced about five times, this is a relative results considering the influence of OS and Matlab.

The calculation accuracy is also held from the analysis of the figures above. In the situation of small step 0.0001, for the 0.1s simulation, 1000 points can be obtained. In the situation of big step size of 0.001, 100 points can be obtained. The value of 100 points are extracted from the situation of small step size, which are the values correspond to the same time points in the situation of big step size. The ratio of calculation error can be achieved by comparing to the values of small step size, which is deemed as precise. The maximum error ratio is calculated by the consideration of $P_{in}, P_{m}$ together. Table 3.2 shows the comparison results of 0.1s calculation.

<table>
<thead>
<tr>
<th>Table 3.2</th>
<th>Results comparison between Common numerical method and proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: Proposed method with inside Common Euler algorithm</td>
<td><strong>B: Common Euler method</strong></td>
</tr>
<tr>
<td>A</td>
<td><strong>Proposed method</strong></td>
</tr>
<tr>
<td><strong>Throttle angle</strong></td>
<td><strong>Integration step size</strong></td>
</tr>
<tr>
<td>84</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>0.0001</td>
</tr>
<tr>
<td>54</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>0.0001</td>
</tr>
<tr>
<td>24</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>0.0001</td>
</tr>
</tbody>
</table>
Figure 3.23 Simulation result use proposed method with inside Common Euler algorithm, step size=0.001

Figure 3.23 is the simulation results under random variable throttle angle inputs same as shown in Figure 3.17 and the trajectory of $P_{in}$ and $P_{m}$ are compared between the proposed method and the common numerical method, which illustrates the error is less then 1% of an acceptable level. The simulation time is one second.

Comparing the simulation results of former numerical method of step size equal to 0.0001 and the proposed method of step size equal to 0.001, it is demonstrated that the calculation speed is greatly improved and the calculation error is restricted to a desirable level. There is still numerical error existing in the proposed algorithm since the numerical integration step is enlarged. From the detail analysis, the numerical error will increase when the throttle angle decreases. This error will have little influence on the control precision in practical since the engine generally works on the medium load. The throttle angle will bigger than 30 degree in this case. Furthermore the error is limited less than 1% in the proposed algorithm, it is an acceptable results for real time control.

3.4.4 Intake System Implementation

There is a compromise of calculation precision and speed of model equations for real time engine control system in current state. In order to increase calculation speed, the mathematic model is greatly simplified; also experiment data and several look-up tables are stored in memory of engine control system. When the sensors get the current sampled data, the corresponding control variables are calculated from experiment data and look-up
tables. The most obvious shortcoming of this method is that the experiment data can't always keep consistent with the sampled data so that the control results can't match the desirable results in the look-up tables. An improved way of this method is to use the linear or non-linear polynomial curve fitting algorithm to replace part of the duty of data matching in look-up tables, but it still can't meet the control requirements of instantaneous change of engine state.

The parameter of air pressure or air flow mass is used as the element for model based predictive control of air-fuel rate. Predictive control refers to the concept where the current control action is based on a prediction of the system output a number of time steps into the future. This means that the next step system output must be got from the intake system model, here is air pressure, and use this output as parameter to decide the quantity of fuel injection. The current parameters and sampled data are obtained; the next step output parameter can be calculated using this internal model. By the predictive output of air pressure and density in the intake manifold the quantity of fuel injection can be decided with the reference of data and parameter from other parts of engine. Therefore the control system performance is mainly determined by the accuracy of internal model, which is embedded into engine ECU.

<table>
<thead>
<tr>
<th>step</th>
<th>Pin</th>
<th>Pm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>97860.0000</td>
<td>97860.0000</td>
</tr>
<tr>
<td>0.0010</td>
<td>98302.4766</td>
<td>97514.3828</td>
</tr>
<tr>
<td>0.0020</td>
<td>98149.9219</td>
<td>97357.5547</td>
</tr>
<tr>
<td>0.0030</td>
<td>98047.0547</td>
<td>97256.2734</td>
</tr>
<tr>
<td>0.0040</td>
<td>97976.4688</td>
<td>97187.2656</td>
</tr>
<tr>
<td>0.0050</td>
<td>97928.0234</td>
<td>97140.3984</td>
</tr>
<tr>
<td>0.0060</td>
<td>97894.7734</td>
<td>97108.7266</td>
</tr>
<tr>
<td>0.0070</td>
<td>97871.9531</td>
<td>97087.4766</td>
</tr>
<tr>
<td>0.0080</td>
<td>97856.2812</td>
<td>97073.3758</td>
</tr>
<tr>
<td>0.0090</td>
<td>97845.5156</td>
<td>97064.1719</td>
</tr>
<tr>
<td>0.0100</td>
<td>97838.1172</td>
<td>97058.3359</td>
</tr>
<tr>
<td>0.0110</td>
<td>97833.0312</td>
<td>97054.8047</td>
</tr>
<tr>
<td>0.0120</td>
<td>97829.5234</td>
<td>97052.8516</td>
</tr>
<tr>
<td>0.0130</td>
<td>97827.1094</td>
<td>97051.9844</td>
</tr>
<tr>
<td>0.0140</td>
<td>97825.4375</td>
<td>97051.8594</td>
</tr>
<tr>
<td>0.0150</td>
<td>97824.2734</td>
<td>97052.2422</td>
</tr>
</tbody>
</table>

3.24a Output of throttle angle equal to 84
3.24b Output of throttle angle equal to 54

3.24c Output of throttle angle equal to 24

This model is implemented on Xilinx FPGA and the outputs on the Hyper Terminal are shown as Figure 3.24, refer to the following sections for detail design procedure. This chip
is embedded into engine electronic control unit (ECU) for model based real time control. The structure of ECU including the FPGA is shown in Figure 3.25. Fuel injector duration control is the most typical control of engine control system. Generally determination of the final injection pulse width is the function of a three step process. The first step evolves calculation of basic injection duration. Input sensors used in basic duration calculation are air flow meter and engine rpm. The ECU calculates basic injection duration based on engine speed and air flow volume. The signal of throttle angle is obtained by ECU and sent to the chip, then the next step manifold pressure, i.e. air flow mass is calculated and transferred back to ECU in the time limitation regulated by engine rpm. The current signal of air flow rate is also achieved by air flow meter and sent to ECU, then the basic fuel injection can be decided by the difference of these two signals and the next fuel injection duration can be predicted. In the second step involves duration corrections by the sensor signal of engine water temperature, intake air temperature and exhaust oxygen content. The final step is a battery voltage correction. Then the final fuel injection can be decided.

Figure 3.25 Embedded system implement
3.5 Summary

A simulation algorithm is proposed for hybrid system—non smooth system consisting of sets of differential equations and discrete parts according to the external control commands and internal evolution rules. In this case the system is not suitable for direct numerical methods and has character of chatting (oscillation) between the intersections of different regions in certain situations. The conditions which trigger the discontinuity, called events, and the continuous parts evolution in respective nodes are precisely detected and efficiently processed. The switching surface is divided to two parts of sliding and crossing, the two modes on the switching surface are analyzed in detail; the control formula and the condition guiding the transition between these two parts are demonstrated. The calculation algorithm is presented to transit the system near this surface of two region into three nodes by extended automation therefore the undesirable transitions are avoided and the trajectories are smooth.

In the end a successful application is illustrated to solve the intake manifold model of automotive turbocharged engine for real time predictive control, in which the stability is increased and the trajectory oscillation is removed greatly. The trajectories are smooth under the entire regions of throttle angle inputs, the calculation speed is increased by eight times compared to the general method and the error is restricted within 1%. This solution is verified on the platform of MATLAB and Visual C++. At last the intake system model is implemented on FPGA chip and it can be embedded into ECU for real time control.
Chapter 4 Anti-lock Brake System Design

4.1 Vehicle Model and Simulation on Software Platform

The basic model is an 8-DOF model of 4-wheel vehicle. The general vehicle model is modified to specify for ABS controller simulation and design, strengthen the brake pipeline model and road model, these two parts have much influence on the performance of ABS controller. The dynamic behavior of a vehicle is introduced, including different parts and forces, which are used to track the motion of the individual vehicle components and the overall frame. Individual forces such as brake pressure, acceleration, and friction with the road surface, tracked and calculated for each wheel individually, determine the overall behavior of the vehicle. Based on the coordinate frames shown in Figure 4.1, the dynamic equations of the vehicle body can be obtained according to the force and moment analysis from the figure above:

![Vehicle body and force model for dynamic simulation](image)

Figure 4.1 Vehicle body and force model for dynamic simulation
\[ M(\dot{V}_x - \phi V_y) = F_{x_1} \cos \delta_1 + F_{x_2} \cos \delta_2 - F_{y_1} \sin \delta_1 - F_{y_2} \sin \delta_2 + F_{x_3} + F_{x_4} \] (4.1)

\[ M(\dot{V}_y + \phi V_x) = F_{x_1} \sin \delta_1 + F_{x_2} \sin \delta_2 + F_{y_1} \cos \delta_1 + F_{y_2} \cos \delta_2 + F_{y_3} + F_{y_4} \] (4.2)

\[ I_x \dot{\phi} = t_f (F_{x_1} \cos \delta_1 - F_{x_2} \cos \delta_2 - F_{y_1} \sin \delta_1 + F_{y_2} \sin \delta_2) + t_r (F_{y_3} + F_{y_4}) + l_f (F_{x_1} \sin \delta_1 + F_{x_2} \sin \delta_2 + F_{y_1} \cos \delta_1 + F_{y_2} \cos \delta_2) - l_r (F_{y_3} + F_{y_4}) \] (4.3)

where, \( M \) is the mass of vehicle, \( \phi \) is yaw rate, \( V_x \) and \( V_y \) denote the longitudinal velocity and lateral velocity, \( F_{x_1} \) and \( F_{y_1} \) denote the longitudinal force and the lateral force generated by the tires respectively, \( I_x \) is the yaw moment of inertia about its mass center, \( l_f \) and \( l_r \) are the distances from the center of mass to the front and rear axles, respectively, and \( t_f \) and \( t_r \) are the halves of the front and rear treads, \( \delta_1 \) and \( \delta_2 \) are the wheel angle of vehicle.

Suppose that the automotive is in idle state when it is braked and the connection between transmission and engine is stiff, then the power train is modeled as a damping sine wave. The amplitude is decided by the character of engine and the frequency is decided by the wheel rotation speed. The engine torque is against to the vehicle motion in brake mode, the resistance force is transferred to tires through transmission, which can be easily modeled as gear conversion system.

The tire force, \( F_{x_1} \) and \( F_{y_1} \) are described as nonlinear functions of the slip ratio, the slip angle, the normal forces and the velocity of the tires. Here the Dugoff model is used to simulate the tire characteristics. The slip angle can be achieved from vehicle body model mentioned above, the slip angle of tire can be achieved as follow:

\[ \alpha_1 = \arctan\left(\frac{V_y + l_f \phi}{V_x - (t_r + t_f) \phi / 2}\right) \] (4.4)

\[ \alpha_2 = \arctan\left(\frac{V_y + l_f \phi}{V_x + (t_r + t_f) \phi / 2}\right) \] (4.5)

\[ \alpha_3 = \arctan\left(\frac{V_y + l_f \phi}{V_x - (t_r + t_f) \phi / 2}\right) \] (4.6)

\[ \alpha_4 = \arctan\left(\frac{V_y + l_f \phi}{V_x + (t_r + t_f) \phi / 2}\right) \] (4.7)

and the slide slip angle \( \beta \) is defined as the angle between the longitudinal axis of the automotive and its local velocity at the center

\[ \beta = \tan^{-1}(V_y / V_x) \] (4.8)

The vehicle speed is not the same as the wheel angular speed when it is braked and
the wheel is being to block. For the front left wheel, the model of wheel angular speed sensor which is used to estimate the wheel angular speed, the main parameter for ABS control algorithm design, is model as:

\[ S = \frac{(V_{fl} - \omega_{fl} \cdot r_{fl})}{V_{fl}} \]  \hspace{1cm} (4.9)

\[ V_{fl} = (V_{x} - (t_{r} + t_{f})\varphi / 2)\cos(\delta_{2}) + (V_{y} + t_{f} \varphi)\sin(\delta_{2}) \]  \hspace{1cm} (4.10)

\[ \dot{\omega}_{fl} = \frac{(F_{c2}R_{fl} - M_{fl})}{J_{fl}} \]  \hspace{1cm} (4.11)

where \( S \) is the slip rate, \( V_{fl} \) and \( \omega_{fl} \) is the line speed and angular speed of front left wheel, \( M_{fl}, J_{fl} \) and \( R_{fl} \) denote the brake moment, angular inertia and rotation radius of the wheel. The parameters of other wheel can be calculated in similar ways.

Figure 4.2 Brake pipeline model and data sample system

The model of brake pipeline and brake are used to generate brake force. Considering that the material and layout of brake pipeline have much influence on the accuracy of its mathematical model, the model is built from the real time sampled data, which is integrated in pure software simulation by curve fitting. Also the hardware structure can be used in HIL and it is flexible for brake system of different kind of vehicles. The structure is shown in Figure 4.2. The brake model can be obtained from the character of kinds of brake or use brake coefficient provided by brake factory directly.

The vehicle state is influenced by the tire-road interaction greatly. The road model is limited in the consideration of friction coefficient on road surface, which provides much information for force and moment loaded on the tire and vehicle. The four kinds of road
model in this system are shown in Figure 4.3:

\[ \mu = \begin{cases} \mu_1 \cdots X \leq X_0 \\ \mu_2 \cdots X > X_0 \end{cases} \]

a. Road model of variable friction coefficient

\[ \mu = \begin{cases} \mu_1 \cdots Y \leq Y_0 \\ \mu_2 \cdots Y > Y_0 \end{cases} \]

b. Road model of separate friction coefficient

\[ \mu = \begin{cases} \mu_1 \cdots [(X - X_0)/a] = odd \\ \mu_2 \cdots [(X - X_0)/a] = even \end{cases} \]

c. Road model of intervein fiction coefficient
d. Road model of chessboard friction coefficient

\[
\begin{align*}
\mu &= \begin{cases} 
\mu_1 \cdots \left[ \frac{(X - X_0)}{a} \right] + \left[ \frac{(Y - Y_0)}{b} \right] = \text{odd} \\
\mu_2 \cdots \left[ \frac{(X - X_0)}{a} \right] + \left[ \frac{(Y - Y_0)}{b} \right] = \text{even}
\end{cases}
\end{align*}
\]

Figure 4.3 Four kinds of road mathematic model with different friction coefficients

In the pure software simulation environment, all the function simulation is implemented by software according to the equations (4.1) to (4.11). Beginning from certain loop of braking, the wheel angular speed is obtained by the model of angular speed sensor, which is the main parameter input to ABS controller. According to the signal of angular speed, through the control algorithm in the controller, output signal, rising pressure, holding pressure, falling pressure or their compromise, is transferred into the model of pipeline system to produce the brake pressure, from which the brake moment is generated by brake model, combining with vehicle body model and road model, the vehicle state of speed and accelerate can be calculated. From the interplaying of vehicle body model, engine model, road model and tire model, the wheel angular speed can be achieved and a new control loop will begin. The following Figure 4.4 and Figure 4.5 show the vehicle body model and front right tire model connected to it in MATLAB/SIMULINK. Other tire models are similar to Figure 4.5. The parameters and output results are stored to database in other computer through Ethernet.
Figure 4.4 Vehicle body model on MATLAB/SIMULINK.

Figure 4.5 Front right of tires model on MATLAB/SIMULINK.
4.2 Hardware in Loop Simulation Based on Distributed Networks

4.2.1 Hardware System Design and Implementation

HIL systems are commonly used for fault tolerant studies (shorted or open signals, etc.) and reliability (endurance) tests of new components after software simulation, for providing more precise data and reference for practical design. HIL simulation evaluates the performance of the ABS ECU by interfacing the ECU to software models of the vehicle, real components of brake pipeline system and interface circuit, and running the system through a variety of brake scenarios and road conditions. The performance of the ABS controller, as well as the software models and hardware system, is monitored by logging and plotting the system variables passed between the different components.

The conventional approaches to vehicle HIL simulation rely on a central computer control system with individual connections to each sensor/actuator, to sample data, produce control command and transfer it. Reconfiguring the vehicle to new component design require extensive rewiring and reprogramming of the central computer system. High cost and lowered reliability result as more subsystems are added. Distributed systems are regarded as a way of increasing system capability while decreasing cost and apparent complexity. In the proposed HIL simulation system, all the components, including software model in computer, wheel rotation modular, engine modular and ABS controller, are connected to CAN bus for simulation the vehicle motion entirely. The modular based distributed system is made by identical and independent mechatronic modular that can be disconnected and reconnected autonomously and rearranged into different structures for different type vehicles to complete its tasks more effectively. Each individual module is a self-constrained unit equipped with its own processor to control the module’s movement and to facilitate communication with other modules.

The basic idea behind this system is to divide the system into individual subsystem composed of sensors, actuators and microcontrollers, according to the actual environments in vehicle. Each of these micro controlled subsystems is called a node and the nodes are connected together by CAN bus. This system is developed to be convenient for reconfiguration of other tests by adding or releasing nodes. By distributing the functionality of the system out of the center computer and into the nodes, the reliability and capability of the overall system increase while the apparent complexity decreases. Here What are considered are the currently-used high speed bus (ISO,1994) connecting engine, ABS, and transmission in vehicle system, the discussion of LIN for lower speed communication and TTCAN without practical application is not considered.

This HIL simulation system is constructed by two computers, several CAN interface
boards and actual ABS control components. The system structure is show as Figure 4.6. The models of brake actuator, tire, road and vehicle body are running in software mode compiled by Visual C++ to control the motion of vehicle. Based on the vehicle dynamics information, the parameters of angular speed of four wheels are transferred to angular speed servo electromotor through CAN bus. The rotation signals are achieved by speed sensors respectively, influence of delay and disturbance is added to these signals, and these signals are transferred to ABS controller and power train model through CAN bus. The delay time and disturbance can be adjusted according to different vehicle types. The ABS controller gets angular speed signals through CAN interface, generating the brake signals (rise, hold, or fall) which are passed to ABS actuators to generate brake pressure in the brake pipeline system. The structure and material of pipeline system have much influence on the character and performance of ABS controller, the parameters can also be adjusted to satisfy different type of vehicles. The pressure value of four brake pipelines are sampled by pressure sensors, the values are transferred back to main computer to generate brake force which will make the vehicle to change its motion states. Also the engine model achieves the rotation speed signal of wheel from CAN bus and makes an average of these four. The engine rotation speed can be transferred back to vehicle dynamic model, generate resistance force to vehicle motion, with the consideration of inertia of power train system.

Figure 4.6 HIL system structure based on distributed network
One computer is used as main control computer, which is connected to CAN bus by CAN interface card PCL-841. The data of angular speed of vehicle model and control command are transferred to CAN bus through this card. Also the real time signals such as brake pipeline pressure are transferred to vehicle model to control the motion of vehicle, from which the performance of ABS controller can be judged. The ABS control command is also transferred to main computer for researching the delay character of pipeline system. PIC12C672 is used as the control MPU for CAN interface circuit, integrated with 4 channels of 8-bit Analog-to-Digital (A/D) converter. Because there isn’t Serial Peripheral Interface (SPI) in this MPU, the SPI interface protocol is simulated using general I/O of GP5, GP4, GP1, GP2, connecting to the pin of CS, SO, SI, SCK. The CAN controller is MCP2510, which supports CAN 1.2, CAN 2.0A, CAN 2.0B Passive, and CAN 2.0B Active versions of protocol, and is capable of transmitting and receiving standard and extended messages. The MPU communication is implemented via SPI with data rates up to 5 Mb/s. The pins of TX0RTS, TX1RTS, TX2RTS is configured as the general inputs, connected to 3-bits switch to set the ID of CAN interface, RX0BF and RX1BF are used as output pins to output control signals for outside devices through driver circuit. The MCP2551 is used as CAN transceiver, it is a high-speed CAN, fault-tolerant device that serves as the interface between a CAN protocol controller and the physical bus. It will operate at speeds of up to 1 Mb/s. The structure of CAN interface is shown in Figure 4.7.
The software model is run on the main compute, and the other computer, connected by Ethernet, is charged for data processing and graphics. Because it is a complex nonlinear dynamic system and the non real time character of Windows, only one work thread is used to execute the calculation, to satisfy the real time requirement of HIL simulation. Suppose that ABS controller and actuator will act 15 times one second, the data exchange with CAN bus, calculation of vehicle dynamics and send data to assistant computer must be completed less than 1/15 second. The important point is vehicle model solution, which is the trajectory of collected differential equations.

When a continuous time system is simulated on a digital computer, its response is computed at closely spaced discrete times. Output results are generated by joining the closely spaced calculated response values with straight line segments, approximating a continuous curve. The accuracy and speed of computation depends on the integration scheme that is used and on the choice of the integration step size. Especially for the real-time simulation of continuous dynamic systems, the choice of numerical integration methods is an important consideration. In real-time simulations, the choice of numerical integration methods is generally based on different considerations than those used in numerical solutions in off-line predictive studies, when there is no real-time requirement. Real time determination and absolutely time control can only be achieved using fixed-time step integration without using iterative solvers. In every integration step pressure signals are got from CAN bus and the current vehicle motion is calculated. Because the pipeline model is specified for ABS application, the entire system becomes very complicated and hard to process. There are three command modes for brake pipeline system, rising, holding and falling, which will cause great vary in the value of pipeline pressure. From the aspect of entire system, this system is very stiff, even discontinuous (Liberzon and Morse 1999, Leine, 2000), at the point of pipeline mode switch and hard to solve, it is easy to introduce bifurcation in the switching point with big integration step size. In the pure software simulation environment, variable step size integration is used to reduce the occurrence of this situation. Considering the real time requirement of this system, the differential equations can’t be solved with small step size. A compromised method is used to solve this problem. The control command of ABS controller is also introduced to the main computer. In each integration step, the signal of control command is monitored. If the signal changed, which means that the pipeline pressure will change greatly, then the integration step size is reduced to 1/10 of former step size, to compensate the stiffness of system and ensure the stability of entire system. Also it is allowed that users can set the integration step size according to different vehicle systems.

By breaking the system up into several small subsystems and subtasks, the function of each given node and hence the software is greatly simplified. Since each node preprocesses and interprets its sensor data, only high level messages need be sent to
other nodes. The internal functionality of each node is hidden; only the communication interfaces need to be known. Thus a given node’s internal code, sensors, actuators may be changed without affecting the way the rest of the network uses the node. Encapsulate makes it much easier to reconfigure systems and interchange components. Furthermore the node components can be standardized. Many different tasks can be accomplished by reconfiguring and reprogramming similar nodes, even removing an unsuitable node and plugging in a new one. This is also the development current for modern vehicles. For the simulation system, this distributed mode exactly agrees with the real physical object and reflects the actual system motions. For ABS designers, this system is very convenient for them to find what and where the trouble begins from monitor the parameters in and out a certain given node respectively.

4.2.2 Data Processing and Interface Design

Visual C++, MATLAB, database and EXCEL is integrated together to make this system automatically and efficiently in the assistant computer. According to the act of ABS system, the time slice defined as 1/15 second. The main computer send data, such as vehicle dynamics, pipeline pressure, to assistant computer in every slice through Ethernet using Socket interface. Database system, data processing and graphic system are constructed in the assistant computer, to share the duty of main computer and ensure the real time calculation of main computer. The date in database is refurbished every 1/15 second, store the current data transferred by main computer. Also the graphic system is also refurbished every 1/15 second to display the vehicle motion state in real time.

The vehicle dynamic parameters of vehicle speed, wheel angular speed, vehicle yaw rate etc., and the hardware parameters such as pipeline pressure, ABS control signals are transferred to assistant computer through Ethernet. The data are stored into database of ACCESS using ADO method (David, 2003), and can be output to EXCEL for analysis, instead of the file system formerly used. The application structure of ADO is shown in Figure 4.8:

The necessary data is stored into database using ADO (ActiveX Data Objects) technology. The classical objects used are Connection object, Command object, Recordset object, Record object, Fields object and Parameter object, which makes it possible for data communication between database and Visual C++. The programming model of ADO is shown in Figure 4.9:
In software simulation, the vehicle parameter such as mass, length, tire parameter, etc. can be taken out from database on assistant computer based on TCP/IP to main computer for computation and processing by MATLAB using COM interface and Add-in technology. MAT-files, the data file format of MATLAB is used for saving data to disk, to import data to and export data from the MATLAB environment, refer to (Stephen, 2002) for details.

MAT-files provide a convenient mechanism for moving MATLAB data between
different platforms in a highly portable manner. The file I/O routines provided by MATLAB are used to complete the data communication between it and Visual C++. Also in this system, COM technology is used to integrate MATLAB into Visual C++ platform. MATLAB supports Automation, which is COM-based technology that enables binding at run time, or late binding, to an object's methods and properties and also makes possible cross-application macro programming. Automation Client is an application, programming tool, or scripting language that accesses services provided by Automation objects; Automation Server is an application, type library, or other source that makes Automation objects available for programming by other applications, programming tools, or scripting languages. In our system, MATLAB is Automation Server, it provides service to Visual C++ that is Automation Client, and therefore Visual C++ can integrate MATLAB into its own environment.

In HIL simulation, the entire system is constructed using Visual C++. The vehicle parameter is also taken out from database on assistant computer using TCP/IP and Socket. These parameters are integrated with the real time data from CAN bus to control the dynamical motion of vehicle. And the calculation results are transferred back to assistant computer for storage and display.

The user interface of parameter configuration is shown as Figure 4.10. ABS designer can set the parameters of different type vehicles and test the ABS performance respectively. The parameters are stored into database. Figure 4.11 is the simulation interface with database. The certain type of vehicle can be selected for executing software simulation or HIL simulation.

![Figure 4.10 User interface of parameter configuration](image)

85
4.3 ABS Controller Design

4.3.1 Fuzzy Control Method

An integrated simulation environment is developed using a combination of computer based platform, distributed architecture, and software solutions to numerical problems for real time system. This system is used for aided design of a fuzzy ABS controller. Some aspects had been introduced by Mark (1995) formerly. Fuzzy logic imitates the logic of human thought, which is much less rigid than the calculations computers generally perform.

For example, consider the task of driving a car. As you drive along, you notice that the stoplight ahead is red and the car in front of you is braking. Your (very rapid) thought process might be something like this: "I see that I need to stop. The road is wet because it's raining. The car is only a short distance in front of me. Therefore, I need to apply significant pressure to the brake pedal immediately." This reasoning takes place subconsciously, of course, but that's the way our brains work-in fuzzy terms. Human brains do not base such decisions on the precise distance to the car ahead or the exact coefficient of friction between the tires and the road, as an embedded computer might. Likewise, our brains do not use a Kalman filter to derive the optimal pressure that should be applied to the brakes at a given moment. Our brains use common-sense rules, which seem to work pretty well. But when finally getting around to pressing the brake pedal, an
exact force is applied. The process of translating the results of fuzzy reasoning to a non-fuzzy action is called defuzzification. This is the principle of fuzzy logic.

The control objective for the ABS controller design is to maintain wheel slip ratio as close as possible to the optimal target value to minimize stopping distance meanwhile ensuring side stability within acceptable range. Since slip ratio is continuous variable, many control theories can be used for its controller design, such as conventional PID or optimal control, etc. Although a PID controller is practical and simple to be implemented, its system parameters need to be adapted corresponding to changing vehicle operation conditions. And the complexity of vehicle system, particular for tire non-linearity, could result in the difficulties in the parameter adaptation. On the contrary, a fuzzy logic controller can easily adapt to the complex, changeable operation condition and nonlinearity existing in the vehicle suspension system and tires, also with good robust performance (Bauer and Tomizuka 1996; Li et al. 2001, Akey 1995). The scheme for the ABS fuzzy logic controller is shown in Figure 4.12.

![ABS fuzzy logic controller diagram](attachment:image)

Figure 4.12 The scheme for the ABS fuzzy logic controller.

For a typical fuzzy controller with two inputs and one output, the control rules \( R_j \) can be expressed as:

\[ R_j : \text{if } \beta_1 \text{ is } A_j \text{ and } \beta_2 \text{ is } B_j \text{, then } \lambda \text{ is } C_j \ (j = 1, 2, ..., n) \]

where \( \lambda \) is output variable, \( A_j, B_j, C_j \) are corresponding input and output fuzzy sets, whose membership functions are determined by \( a_1, a_2, a_{el}, \) and \( a_{e2} \), respectively, and now presumed to be identical and shown in Figure 4.13.
The triangle membership function has been widely used in fuzzy controller design (Li et al., 1998; Chou et al., 1998; Yoshimura et al., 1997; Yu et al., 2001). The design procedure of the fuzzy logic controller includes three main parts, i.e., fuzzification, fuzzy reasoning, defuzzification. Each part is respectively presented in detail as following.

(1) Fuzzification

The fuzzification process of input variable can be expressed by following equations,

\[ \lambda_{i,j} = \mu_{A_i}(\beta_i), \lambda_{z,j} = \mu_{B_j}(\beta_z) \]  \hspace{1cm} (4.12)

where \( \mu_{A_i}(\cdot) \) and \( \mu_{B_j}(\cdot) \) respectively represent the membership functions of \( A_i \) and \( B_j \). \( \lambda_{i,j} \) and \( \lambda_{z,j} \) are degrees of fitness of input value \( \beta_i \) and \( \beta_z \), i.e., measurements of input variable \( \beta_i \) and \( \beta_z \).

(2) Fuzzy reasoning

In present study, Mandain method is used in fuzzy reasoning process, given as,
\[ \lambda_j = \lambda_{ij} / \lambda_{2j} \]  

(4.13)

where \( \lambda_j \) denotes the degree of fitness of fuzzy reasoning results, and symbol \( \wedge \) conducts minimum operation. In each control rule \( r_j \), the membership function for output variable \( Y \) can be expressed as \( \lambda_j \mu_{c_j}(y) \), in which \( \mu_{c_j}(\cdot) \) denotes the membership function of \( c_j \).

(3) Defuzzification

For the improvement of reasoning and computation speed, the product-sum gravity method (Kandel and Langholz, 1993) is used to obtain the defuzzified value of output variable \( Y \). The product-sum-gravity method gives the defuzzified value \( y^0 \) is,

\[ y^0 = \frac{\sum \gamma_j \lambda_j S_{c_j} / \sum \lambda_j S_{c_j}}{\sum \gamma_j \lambda_j S_{c_j} / \sum \lambda_j S_{c_j}} \]  

(4.14)

where \( S_{c_j} \) is area of \( \mu_{c_j}(\gamma) \), \( \gamma_j \) is distance from center of gravity of \( S_{c_j} \) to the base point of zero.

The empirical knowledge is used to construct the fuzzy control rules for suspension control system. And the fuzzy rules are described by language values as below (Chou et al., 1998; Yoshimura et al., 1997).

(a) If and are both positive (negative) big, then \( Y \) negative (positive) big;
(b) If the absolute value of and is both small, then the absolute value of \( Y \) is relatively small;
(c) If is positive big (negative big) and is negative big (positive big), then the absolute value of \( Y \) is small;
(d) If is positive big (negative big), then \( Y \) is negative (positive) medium;
(e) If the absolute of value is small, then \( Y \) is small.

To summarize, the above fuzzy rules expressed by language values are shown in Table 4.1.

<table>
<thead>
<tr>
<th>( g )</th>
<th>( 1B )</th>
<th>( 2B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>NB</td>
<td>NB</td>
</tr>
<tr>
<td>NS</td>
<td>PB</td>
<td>PB</td>
</tr>
<tr>
<td>ZO</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>PS</td>
<td>PS</td>
<td>PS</td>
</tr>
<tr>
<td>PB</td>
<td>PS</td>
<td>PS</td>
</tr>
<tr>
<td>PB</td>
<td>ZO</td>
<td>ZO</td>
</tr>
<tr>
<td>NS</td>
<td>ZO</td>
<td>ZO</td>
</tr>
<tr>
<td>PS</td>
<td>PS</td>
<td>ZO</td>
</tr>
<tr>
<td>PB</td>
<td>ZO</td>
<td>PS</td>
</tr>
<tr>
<td>ZO</td>
<td>ZO</td>
<td>ZO</td>
</tr>
<tr>
<td>PS</td>
<td>PS</td>
<td>PS</td>
</tr>
<tr>
<td>PB</td>
<td>ZO</td>
<td>NS</td>
</tr>
<tr>
<td>NS</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>PB</td>
<td>NB</td>
<td>NB</td>
</tr>
</tbody>
</table>

Table 4.1 Fuzzy rules
4.3.2 Controller Design and Simulation

The fuzzy control of ABS controller is depicted as the following. One of the inputs to the controller is the angular speed of vehicle wheels, from which the calculation slip rate can be got. The other input parameter is target slip rate, which is achieved from variable road experiments for maximum friction force and manipulate stability. The slip rate control (John, 1999) is the most important element in ABS control algorithm design. The error of calculation slip rate and target slip rate is used to complete the control logic design. The error between calculation slip rate and target slip rate and its differential are shown as:

\[ \lambda_{\text{error}} = \lambda - \lambda_{\text{target}} \]  \hspace{1cm} (4.15)

\[ \lambda'_{\text{error}} \approx \Delta \lambda_{\text{error}} / \Delta t \]  \hspace{1cm} (4.16)

The output of controller is desirable pressure variable \( \Delta p \), therefore the pressure in pipeline can be calculated as:

\[ p = p + \Delta p \]  \hspace{1cm} (4.17)

Combining the typical fuzzy logic controller design method and the fuzzy control ABS model, based on equations (4.15) to (4.16), the membership functions for input and output variables are defined in Figure 4.14. In this study the Mandain algorithm is used for fuzzy logic operation. Here, \( \lambda_{\text{error}} \) is corresponding to \( \beta_1 \), and \( \lambda'_{\text{error}} \) is corresponding to \( \beta_2 \). They are both input variables. \( \Delta p \) is corresponding to \( \lambda \) as an output variable.

![Figure 4.14a Membership function for error of slip rate \( \lambda_{\text{error}} \)](image-url)
In the stage of defuzzification, based on our empirical knowledge, the Gravity algorithm is used for anti-fuzzilization calculation. The fuzzy rules are expressed by language values are shown in Table 4.2, which shows a set of fuzzy rules that might be used for the ABS controller. It can be used to calculate the output pressure of break pipeline system. The controller inputs are the variables error of slip rate and its differential. The output represents a desirable pressure variable $\Delta p$.

Table 4.2 Rule table for ABS fuzzy control logic

<table>
<thead>
<tr>
<th>$\lambda_\infty$</th>
<th>PB</th>
<th>PS</th>
<th>ZO</th>
<th>NS</th>
<th>NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB</td>
<td>NB</td>
<td>NB</td>
<td>NB</td>
<td>NS</td>
<td>ZO</td>
</tr>
<tr>
<td>PS</td>
<td>NB</td>
<td>NB</td>
<td>NS</td>
<td>PS</td>
<td>PS</td>
</tr>
<tr>
<td>ZO</td>
<td>NB</td>
<td>NS</td>
<td>ZO</td>
<td>PS</td>
<td>PB</td>
</tr>
<tr>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>ZO</td>
<td>PB</td>
<td>PB</td>
</tr>
<tr>
<td>NB</td>
<td>NS</td>
<td>NS</td>
<td>PS</td>
<td>PB</td>
<td>PB</td>
</tr>
</tbody>
</table>

where PB: Positive Big; PS: Positive Small; ZO: Zero; NS: Negative Small; NB: Negative Big.

According to the software models, hardware devices, interface circuit, the outputs
most suitable to evaluate the performance of ABS controller, vehicle speed, wheel speed, slip rate and pipeline pressure, under the control of fuzzy controller on different road conditions are shown in Figure 4.15 and Figure 4.16 respectively.

Figure 4.15 Parameter output on the road model of high friction coefficient
Nation Instruments also provides a real time simulation platform for ABS design. The brake actuator model and vehicle dynamics are running in software on a PXI controller.

Figure 4.16 Parameter output on the road model of high to low friction coefficient.
The brake actuator model is implemented in LabVIEW code, while the vehicle dynamics are modeled and running in the CarSim software, which is integrated into the LabVIEW application. The communication and signal interface between the PXI system and the physical ECU controller is implemented using National Instruments measurement hardware, specifically the PXI-7831R reconfigurable I/O (RIO) card. This system is difficult to expend and reconfigure to integrate necessary vehicle components, specifying road surface condition and brake pipeline system for specified ABS design. Probably impossible be reconstructed for other vehicle applications. Figure 4.19 shows plots of the vehicle velocities and brake pressures as simulated in the ABS ECU test application. In these figures, the abrupt change of wheel speed and the delay of time tag can’t be expressed well. It is concluded that our proposed system constructs a much more real vehicle simulation platform, more close to the actual vehicle environment and more suitable for ABS design.

Figure 4.19 Vehicle velocities and brake pressures simulation using LabVIEW
4.4 Summary

This chapter explores a novel method of building a virtual vehicle system, integrating pure software simulation and hardware in loop (HIL) simulation which is constructed based on distributed network of CAN and TCP/IP. First the dynamic vehicle model is developed in the platform of software simulation, used to design and verify the response of ABS control logic; Second in HIL simulation the model equations are converted to a stand-alone program of C++ on the main computer and integrated with ABS controller, real vehicle devices and components, generating a real vehicle environment based on CAN bus to verify the control performance of ABS in real time. Database, graphic, and data processing system are integrated by cross-platform programming technology on assistant computer connected by Ethernet. In the end the control logic of ABS controller is developed using fuzzy control algorithm. The analysis and validation of developed control logic for ABS can be performed successfully on this platform.
Chapter 5 Automotive Security System Design

5.1 Elements of Independent Component Analysis

5.1.1 Introduction of ICA

Recently, several methods have been proposed to learn image codes that utilize a set of linear basis functions. These approaches have in common that they try to reduce the information redundancy by capturing the statistical structure in images that is beyond second order information. Among these methods, independent component analysis (ICA), as a statistical and computational technique for revealing hidden factors that underlie in sets of random variables, measurements, or signals, is widely used for image processing such as image feature extraction, image recognition and so on.

ICA was originally developed to deal with problems that are closely related to the cocktail-party problem. Since the recent increase of interest in ICA, it has become clear that this principle has a lot of other interesting applications as well. For example, some interesting information on brain activity can be revealed by giving access to its independent components. Moreover, finding underlying independent causes is a central concern in the social sciences. Another, very different application of ICA is feature extraction. A fundamental problem in signal processing is to find suitable representations for image, audio or other kind of data for tasks like compression and denoising. Data representations are often based on linear transformations. Standard linear transformations widely used in image processing are, for example, the Fourier, Haar, and cosine transforms. Each of them has its own favorable properties. It would be most useful to estimate the linear transformation from the data itself, in which case the transform would be ideally adapted to the kind of data that is being processed. In this chapter, feature extraction by ICA for the security system will be explained in detail. To illustrate it clearly, the definition and operating procedure of ICA will be introduced first.

To rigorously define ICA, a statistical “latent variables” model can be used. It is observed $n$ random variables $x_1, \ldots, x_n$ are modeled as linear combinations of $n$ random variables $s_1, \ldots, s_n$:

$$x_i = a_{i1}s_1 + a_{i2}s_2 + \ldots + a_{in}s_n, \text{ for all } i = 1, \ldots, n$$  \hspace{1cm} (5.1)
where the \( a_{ij} \), \( i, j = 1, \ldots, n \) are some real coefficients. By definition, the \( s_i \) are statistically mutually independent.

This is the basic model. The ICA model is a generative model, which means that it describes how the observed data are generated by a process of mixing the components \( s_j \). The independent components \( s_j \) are latent variables, meaning that they cannot be directly observed. Also, the mixing coefficients \( a_{ij} \) are assumed to be unknown. All that are observed are the random variables \( x_i \), and it is necessary to estimate both the mixing coefficients \( a_{ij} \) and the ICs \( s_j \) using the \( x_j \). This must be done under as general assumptions as possible.

ICA is very closely related to the method called blind source separation (BSS) or blind signal separation. A “source” means here an original signal, i.e., independent component, like speaker in the cocktail-party problem. “Blind” means that we know very little, if anything, of the mixing matrix, and make very weak assumptions on the source signals. ICA is one method, perhaps the most widely used, for performance blind source separation.

### 5.1.2 Modeling Procedure of ICA

In the process of ICA calculation, ICA definition is given first. Assume that \( n \) mixed signals are denoted by column vector \( X \) whose elements are denoted by \( x_1, x_2, \ldots, x_n \) and the independent components of column vector are defined as \( S \) with elements \( s_1, s_2, \ldots, s_n \). Also define \( A \) with elements \( a_{ij} \) as mixing matrix. The above model is written as:

\[
X = AS
\]  

ICA is originally proposed to solve the blind source separation problem, to recover \( n \) original source signals \( S \) from the linearly mixed signal \( X \), while assuming as little as possible about the natures of mixing matrix \( A \). By defining the matrix \( W = A^{-1} \), convert the model according to (5.2) as:

\[
S = WX
\]  

So that the independent components can be calculated from the linear transformation by \( W \).

PCA (principal component analysis) (Pajunen, 1998) is a preprocessing step to ICA which reduces the redundancy of mixed signals on a certain level. In the PCA transformation the vector \( X \) is first centered by subtracting its mean then \( E\{X\} = 0 \). Next find a rotated orthogonal coordinate system such that the elements of \( X \) in the new coordinates become uncorrelated and \( X \) is linearly transformed to another vector \( Y \).
with \( m \) elements \( m < n \), so that the redundancy induced by the correlation is removed.

In the PCA by variance maximization define that

\[
C_X = E\{XX^T\} \tag{5.4}
\]

is the \( n \times n \) covariance matrix of \( X \) and \( e_1, e_2, \ldots, e_n \) are the unit-length eigenvectors of the matrix. The eigenvectors are ordered according to the ordering eigenvalues \( d_1, d_2, \ldots, d_n \) satisfying \( d_1 > d_2, \ldots, d_n \). Then \( m \) dimensional vector \( Y \) is:

\[
Y = EX \tag{5.5}
\]

got by choosing the first \( m \) eigenvectors of \( e_i (i = 1 \ldots m) \) as the row vector of \( E \), refer (Tipping and Bishop, 1999) for details.

After reduce the redundancy of mixed signals, the ICA problem can be greatly simplified if the signals are first whitened (Ruymagaart and Robust, 1981) with linear transformation of

\[
Z = VY \tag{5.6}
\]

Define \( V = D^{1/2}E^T \), where \( E = (e_1, e_2, \ldots, e_n) \) whose columns are the \( m \) eigenvectors of \( C \), which has the same mean as in (5.4) and \( D = \text{diag}(d_1, d_2, \ldots, d_n) \) is the diagonal matrix of the eigenvalues of \( C \). Recalling that \( C \) can be written as \( C = EE^T \), with \( E \) an orthogonal matrix satisfying \( E^TE = EE^T = I \), it holds:

\[
E\{ZZ^T\} = VE\{XX^T\}V^T = D^{1/2}E^TVEDE^TED^{1/2} = I \tag{5.7}
\]

then it is demonstrated that \( Y \) is whitened by \( V \).

The fixed-point algorithm (Remagnino et al., 1998; Wren et al., 1997) is realized using C++ and Matlab to find the independent components of whitened vector \( Z \). Suppose a transformation as (5.3) is defined as:

\[
S = WZ \tag{5.8}
\]

where \( S = [s_1, s_2, \ldots, s_m] \) is the vector of independent component. Define the mutual information

\[
J(s_1, s_2, \ldots, s_m) = J(S) - \sum J(S_i) \tag{5.9}
\]
of \( S \) which is a measurement of the independence of random variables, \( J(S) \) is negentropy given by \( J(S) = H(S) - H(S_{\text{gauss}}) \), \( S_{\text{gauss}} \) is a Gaussian random variable of the same covariance matrix as \( S \), and \( H \) is the differential entropy (Sumpter and Bulpitt, 2000). To get the desirable matrix \( W \) from maximizing the mutual information (5.9) of independent component \( S \), maximize

\[
J(s_i) = \left[ \mathbb{E}[G(W^T Z)] - \mathbb{E}[G(v)] \right]^2
\]

(5.10)

under the constrain of \( \mathbb{E}[G(W^T Z)^2] = 1 \) with appropriate function \( G \). The final result of \( W \) is got by Newton iteration of

\[
W^* = W - \mu \left[ \mathbb{E}[Z g(W^T Z)] - \beta W \right] \left[ \mathbb{E}[g'(W^T Z)] - \beta \right]
\]

\[
W = W^* \cdot \left\| W^* \right\|^{-1}
\]

(5.11)

where \( \beta \) can be approximated as \( \mathbb{E}[Z^T Z g(W^T Z)] \), and \( \mu \) is a step size parameter that make the solution converge faster and \( g \) is derivative function of \( G \) defined in (5.10), which decide the converge speed of ICA algorithm (Toyama et al., 1999). For several units of mixed signals, the output matrix \( W_u = (W, Z, W, Z, ..., W, Z) \) must be decorrelated after every iteration to prevent the units from converging to the same maximum, following the iterative algorithm:

\[
W_u = W_u \left( W_u C W_u^{-1} \right)^{1/2}
\]

\[
W_u = \frac{2}{3} W_u - \frac{1}{2} W_u C W_u^{-1} W_u
\]

(5.12)

Then the matrix according to (5.11) is symmetrically de-correlated, see (Amari et al., 1996) for details.

### 5.2 Feature Points Extraction using ICA

After analyze algorithm of ICA, ICA is use to perform redundancy reduction for the input data, i.e a representation using independent components for original image. Field (Field, 1994) has argued that oriented features constitute a sparse representation of the images. This means that for one image, only a few features are needed to represent it. Note that when establish a firm connection between ICA results and signals will be processed, it is need to set the input images to be as close as possible to those which ICA would receive as input. Luckily, it seems that the ICA basis is not very sensitive to the particular set of images used, as earlier work on ICA for feature extraction of images has given
qualitatively quite similar features using different data sets (Bell and Sejnowski, 1997; Hateren and Ruderman, 1998). Thus, the selection of a reasonable dataset input of the common images for ICA can satisfy the requirement of its nongaussian distribution.

Before ICA processing, the image sizes are normalized, both the shot images and the images in the database. In order to adapt to different type of camera and color content, each pixel (RGB-triple) is projected onto a plane by average RGB values. In other words, the influence of luminance and color are ignored, only the element image is used in this system. Image representations are often based on discrete linear transformations of the observed data. Consider a image whose gray scale value at the pixel indexed by \( j \) and \( k \) is denoted by \( I(j,k) \). Here the basic models in image processing express the image pixel \( I(j,k) \) as a linear superposition of some features \( s_i(j,k) \):

\[
I(j,k) = \sum_{i=1}^{n} a_i s_i(j,k)
\]  

(5.13)

where the \( a_i \) are coefficients different for image pixel \( I(j,k) \). Alternatively, it is possible to just collect all the pixel values in a single vector \( X = (x_1, x_2, \ldots, x_m)^T \) where \( m \) denotes pixel number in one image, in which case it is possible to express the representation as \( X = AS \) just like basic ICA. Every element \( x_i \) in \( X \) is represented by the product of row vector in coefficient matrix \( A \) and column components \( S \), therefore the components \( S \) can be considered as the compatible representation of observed variables \( X \). In general there is no need that the number of independent components equals the number of observed variables. It is assumed here that the independent components are much less than the number of observed variables, therefore the data redundancy is greatly reduced.

Considering the system speed and precision requirement, it is not necessary to model a whole image using equation (5.13) in practice. Rather, it is applied to image blocks or patches. Thus the image is partitioned into blocks, and each block is expressed with equation (5.13). Suppose one image is divided into \( k \) blocks, each block is digitized as a column of mixed matrix \( X \), and suppose there are \( l \) images should be processed, then \( X \) is a matrix of column \( k \times l \), and the row of \( X \) equals to the pixel number in every block, named \( t \). The \( u \) feature points of every image block are extracted using ICA algorithm and these feature points are composed together to achieve the feature vector of the initial image. The element number of feature vector is \( m = k \times u \) for one image.

An example of 4-division images of 100*100 is shown as Figure 5.1. Suppose there are 30 images, all images are divided into 4 blocks. In this case, the first 4 columns of matrix \( X \) marked with superscript 1 are the discrete date of the first image, and one column is correspondent to one block. The row number of \( X \) is the pixel number in one
image block \((i=50\times50)\) and the column number is the number of divided image part multiply the image number \((k\times l=4\times30)\). Therefore the images are abstracted to matrix \(X\) consistent with ICA model of equation (5.2). Similarly, the first 4 columns in matrix \(S\) are the feature points of original image blocks abstracted by ICA respectively, here it is set that \(u=16\). These data can represent the original image after be integrated to one feature vector, here is \(k\times u=4\times16\), and the data redundancy is greatly reduced \((u<<t)\). Feature vectors of some images are shown as Figure 5.2, the vertical axes is the value of feature vector elements and horizontal axes is the number of feature vector elements.

\[
\begin{bmatrix}
X_{11} & \cdots & X_{1u} \\
X_{21} & \cdots & X_{2u} \\
\vdots & \ddots & \vdots \\
X_{n1} & \cdots & X_{nu}
\end{bmatrix} = A \times 
\begin{bmatrix}
S_{11} & \cdots & S_{1u} \\
S_{21} & \cdots & S_{2u} \\
\vdots & \ddots & \vdots \\
S_{n1} & \cdots & S_{nu}
\end{bmatrix}
\]  

(5.14)

Figure 5.1 Image modeling for ICA analysis

Figure 5.2 Feature vectors extracted by ICA
The feature vectors of shot image can also be extracted. The shot images must be preprocessed and divided into blocks same as images in the database exactly. Suppose shot images are achieved and also divided the image into 4 parts, the feature points can be extracted as equation (5.15), where $W$, called pattern matrix, is the matrix got in ICA procedure for images in database.

\[
\begin{bmatrix}
S_{11}^1 & \cdots & S_{14}^4 \\
S_{21}^1 & \cdots & S_{24}^4 \\
\vdots & \ddots & \vdots \\
S_{n1}^1 & \cdots & S_{n4}^4 \\
\end{bmatrix}
= W \times
\begin{bmatrix}
X_{11}^1 & \cdots & X_{14}^4 \\
X_{21}^1 & \cdots & X_{24}^4 \\
\vdots & \ddots & \vdots \\
X_{n1}^1 & \cdots & X_{n4}^4 \\
\end{bmatrix}
\]  

The feature vectors can be achieved by integrate the feature points of 4 parts of $S$ respectively. Suppose there are $n$ images in the database and $a$ shot images. The feature vectors of database image and the shot image should be $n$ and $a$. The feature vectors of shot images and the images in the database are constructed to form a matrix of $m \times (n + a)$, where $m$ means the number of feature points in one image feature vector. The target is to select an array of images which are most similar to the shot images from the database to detect what kind of motion pattern the shot images are, i.e. normal or abnormal.

5.3 Motion Recognition and Match Analysis

5.3.1 Cluster Images in Database

Once feature vectors are extracted from the images, it is need to compare them and try to establish a correspondence. When the number of objects in the database is small, it is acceptable to model matching by sequentially examining each image in turn. And it can achieve the possible solutions for which there exists a correspondence of image features. The time consumption of comparison needed to identify an object grows rapidly with the number of shot images and images in the database increasing. Therefore sequential examination of feature matching is not suitable for problems involving large library of images.

The images in database are classified before matching analysis instead of direct sequential comparison. Classification (Jain, 1999) or cluster includes a broad range of decision-theoretic approaches to the identification of images. The classification algorithm is based on the assumption that the images depict one or more features which are abstracted by ICA and that each of these features belongs to one of several distinct and
exclusive classes. The classes may be specified a priori by an analyst (as in supervised classification) or automatically clustered (i.e. as in unsupervised classification) into sets of prototype classes. In the latter the analyst merely specifies the number of desired categories. The numerical properties of various image features are analyzed and the data is organized into categories in order to improve the precision and decrease time consumption. In the procedure of abnormal pattern recognition, the basic cluster is specified in advance and the program is made to identify the remained images into basic cluster automatically. The classification algorithm employs two phases of processing: training and testing. In the initial training phase, characteristic properties of typical images are isolated and a unique description of each classification category, i.e. training class, is created. In the subsequent testing phase, these training classes are used to classify remained images and to construct the clusters.

Improved fuzzy C-mean algorithm (Hoppen et al., 1999) is used to divide the image data into different groups or clusters after initial clusters are selected by human understanding and vision mechanism. Fuzzy cluster allows feature vectors to have membership of multiple clusters based on varying membership degrees, considering a data point which is close to both clusters. Fuzzy cluster copes with such characters and assigns the feature vectors equal memberships to both clusters. This character is exactly suitable for our system. The fuzzy C-means aims to minimize an objective function such as below:

$$J = \sum_{j=1}^{c} \sum_{i=1}^{n} u_{ij}^{m} \|x_{i}^{(j)} - c_{j}\|^2$$  \tag{5.16}

where \(\|x_{i}^{(j)} - c_{j}\|\) is a chosen distance measure between a data point \(x_{i}^{(j)}\) and the cluster centroid \(c_{j}\), also it is the most common Minkowski distance metrics of Euclidean distance \(r=2\). Furthermore the user can choose other distance measure functions according the features of image and the experiment results. The elements \(u_{ij}\) of the membership matrix \(U\) represent the degree of membership, in the range \([0,1]\), of a feature vector \(x_{i}\) to the fuzzy cluster \(c_{j}\). The fuzzifier \(m\) determines the level of cluster fuzziness. A large \(m\) results in smaller memberships \(u_{ij}\) and hence, fuzzier clusters. In the limit \(m=1\), the memberships \(u_{ij}\) converge to 0 or 1, a crisp partitioning. \(m\) is commonly set according to the experimentation or domain knowledge. The detail steps can be shown as follows:

1. Confirm the number of clusters, \(c\) known as \(2 \leq c < n\) of motions;
2. Choose an appropriate level of cluster fuzziness, \(m \geq 1\);
3. Initialize the \((n \times c)\) sized membership matrix \(U\) to random values such that \(u_{ij} \in [0,1]\) and \(\sum_{j=1}^{c} u_{ij} = 1\);
4. Calculate the cluster centroid \( c_j \) using
\[
\begin{align*}
    c_j = \frac{\sum_{i=1}^{n} (u_i)^{m} x_i}{\sum (u_i)^{m}}, \text{ for } j = 1, 2, ..., c;
\end{align*}
\]

5. Calculate the distance measure \( d_{ij} = \| x_i - c_j \| \), for all cluster \( j = 1, 2, ..., c \) and data points \( i = 1, 2, ..., n \)

6. Calculate the fuzzy membership matrix \( U \) according to \( d_{ij} \). If \( d_{ij} > 0 \) then
\[
    u_{ij} = \left[ \sum \left( \frac{d_{ij}}{d_{a_{ij}}} \right)^{\frac{1}{m}} \right]^{-1}.
\]

If \( d_{ij} = 0 \) then the data point \( x_j \) coincides with the cluster centroid \( c_j \), and so full membership can be set \( u_{ij} = 1 \).

7. Repeat from 4 until the change in \( U \) is less than a given tolerance.

For great image database, first it is necessary to classify the images in it to decrease time consumption of matching in real time. Because the feature vectors are abstracted from original images exactly, they are representation of original images. This means that these feature vectors can be processed just as original images. Suppose that feature matrix \( (s_1, s_2, ..., s_n) \) are column vectors in \( S \), where \( n \) is the number of images in the database. This representation allows us to consider each image feature vector as occupying a point, and each training class as occupying a sub-space (i.e. a representative point surrounded by some spread, or deviation), within the \( n \)-dimensional classification space. Therefore the classification problem is that of determining to which sub-space classes each feature vector belongs.

In the cluster routine, two of input parameters are the feature vectors to be classified and predefined clusters. These predefined clusters are initial cluster centroid positions. They are decided by observers and the feature of these motions can be used as initial knowledge to extract class descriptors. In the example database of Figure 5.3 of abnormal actions around the door of car, the predefined cluster can be selected and marked [1] to [7]. The class descriptors should have significantly different descriptions between different groups (clusters) and share the common definitive descriptions in the same group. For the beginning, these predefined clusters are valued as \( c_j \) in equation (5.16) and the feature vector is valued as \( x_i^{(n)} \). After the loop of cluster algorithm finished, the image are separated to different clusters and an array value of new \( c_j \) can be obtained as the cluster centroid, which can be used for the first step in matching analysis. The return parameters are 2-D array: index of the feature vector and the cluster number it belong to.

In the example database of Figure 5.3, the return value of 5th image should be \([5,1]\), 1st 5 is its index in database and 2nd 1 is its cluster number after classification. The
classification result of image is shown in Figure 5.4 and these images are classified to 7 clusters.

Figure 5.3 Image data in the database before cluster

Figure 5.4 Image data after cluster for matching
The image data in database is divided into different clusters with unique centroid $c_j$, which is used as the criterion to discriminate the cluster of real time images shot by camera. The feature vectors of shot images are extracted. By calculating the distance between this array of images and the cluster centroids using the same distance measurement in cluster algorithm, it can be decided which cluster a certain image belongs to.

5.3.2 Recognize and Match Shot Images

The feature vectors are the representation of original images, also they are independent components after ICA. The character of these values is analyzed and many experiments are done for matching analysis. A proposed similarity calculation method is developed to recognize the most similar image from the certain cluster according to the extracted feature vectors. In practice, the feature points of image block will have some peak values to denote the motions. These are presented by weight of feature points after sort. Then these special areas can be focused to detect the entire motion. The proposed method strengthens the influence of peak values using sorted denominator and is suitable for the data achieved by ICA. Also this algorithm partly replaces calculation of floating point number with integer and increases the speed for real time systems.

The $M$ maximum absolute values and index are used to calculate the similarity.

There are two parts in this algorithm. First part is the creation of factor table of images in the database, second part is to calculate the similarity of a shot image and a database image. The steps are shown as below and the example is shown as Figure 5.5.

![Figure 5.5 Explanation of similarity calculation of images](image-url)
First Part: For an image in the database, a factor table is made:

Step1: Get the feature points number \( u \) of one image block. Here \( u = 20 \). Select \( M \) feature points in this block according to the sort of maximum absolute value as it is show in Part A1, which is compromise of precision and execution time. Here define \( M = 3 \).

Step2: Define a 2-D array to record the feature factors of this image, it is called factor table as shown in Part A2. The factor table includes the index (place) and the sign of feature points. Here 3 points are used: \( m = 1,2,3 \), and the index equal to 9, 20 and 1. All the signs of these three points are negative.

Second Part: For a shot image, the similarity is calculated:

Step3: The index and feature points of a shot image block is defined as in Part B. Sort the values according to maximum absolute value then sorted points can be obtained as Part C. The first column of Part C shows the weight of sorted feature points. Here these weight is denoted by \( x \).

Step4: Get the index from the factor table. Using this index, get the feature points from Part B. Then the points weight can be obtained in Part C according to the feature points. Calculate the similarity according to

\[
similarity_i = \prod_{m=1}^{M} \frac{\mu \times x}{u - m + 1}
\]  

(5.17)

\( i \) is the block number and \( \mu \) is the mark of sign agreement. In this example, (1) if \( m = 1 \) and the index equal to 9, it is possible to get the feature point equal to -1.935400 from Part B, and then we can get the weigh of this feature point \( x = 20 \). After that, check the sigh of feature point, \( \mu = 1 \) if the sign of feature points of shot image are same as which in the factor table, \( \mu = 0 \) if opposite. Then it is possible to calculate the \( similarity_{1a} = 1 \times 20 / (20 - 1 + 1) \) . Process the next step of \( m = m + 1 \) and calculate the similarity rate according to (5.17) till \( m = M \). (2) if \( m = 2 \), \( similarity_{12} = 1 \times 9 / (20 - 2 + 1) \) , and (3) for the case \( m = 3 \), \( similarity_{13} = 0 \times 4 / (20 - 3 + 1) \) . Then the final similarity of the block of database image and shot image is \( similarity_{1} = 1 \times (9 / 19) \times 0 = 0 \) . This means these two blocks are not similar.

If the image is divided to blocks of number \( D \), the similarity is accumulated following as:

\[
Similarity' = \sum_{i=1}^{D} similarity_i
\]  

(5.18)

After the similarity of all image blocks are calculated, the similarity of one image can be achieved according to (5.18). Therefore the most similar ones in database can be
decided. The bigger similarity means more similar. This means the shot image is recognized as similar to the database image.

The calculation time is reduced and precision is improved using the method above comparing with minimum mean-square error method. An image is usually divided into 9 parts according to the requirement of real-time processing. Table 1 shows the comparing result of successfully recognizing the desirable images of the two methods on the same software conditions.

Table 5.1 Comparison of the two recognition methods

<table>
<thead>
<tr>
<th>Calculation of similarity</th>
<th>Number of division</th>
<th>Rate of recognition (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum mean method</td>
<td>1</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>46</td>
</tr>
<tr>
<td>Proposed recognition method</td>
<td>1</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>57</td>
</tr>
</tbody>
</table>

Also the influence of division number of an image to the successful recognition rate is tested and recommended in the software environment before implement to hardware system. A set of calculation efficiency is also tested according to the division number. Figure 5.6 shows the relationship of different division number, feature points and recognition rate.

Figure 5.6 Relationship of different division number feature points and recognition rate
Before extracting the feature points using ICA method, the image can be preprocessed to remove unnecessary section, rid noise and regular size. The rate of effective feature points is increased and the recognition rate is also improved. Table 2 gives the results of recognition rate with preprocessing using proposed recognition method under different division number.

Table 5.2 Result of recognition rate with preprocessing

<table>
<thead>
<tr>
<th>Number of division</th>
<th>Number of feature points</th>
<th>Rate of recognition(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>4</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>59</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>36</td>
</tr>
</tbody>
</table>

Suppose each image in the database is valued an abnormal weight \( v_i \), \( i = 1, 2, \ldots, n \) from system requirements and opinions of observers. The bigger value means the more abnormal motion. The \( s \) shot images are ranged into \( c \) clusters, the number of image in cluster \( c_p \) is defined by \( s_q \), \( p = 1, 2, \ldots, c \) and \( q = 1, 2, \ldots, s \), the abnormal coefficient can be calculated as:

\[
N = \sum_{j=1}^{c} \sum_{i=1}^{s} v_i
\]  

(5.19)

If this coefficient is greater than the pre-defined value, these motions are abnormal motions. The pre-defined value is decided from many experiments in local places and the experience of the observers. For example, get the mean of abnormal weights for every row of images just like in Figure 5.4, and define the sum of means as a pre-defined value. If most of shot images are like the first row of image database, the abnormal coefficient must be less than the pre-defined value, otherwise, if most of shot images like the 5th row, the abnormal coefficient must be greater than the pre-defined value, therefore these motions are considered as abnormal. The definition of the pre-defined value can be precise enough in practice according to the experiments.

An example of 40 shot images of four people shot by digital camera is given. The motions of these four people are divided to eight arrays respectively, marked as array 1 to array 8 from top to lower. In every array five shot images are used to show the motion modes. It is shown in Figure 5.7.
Define the abnormal weight of images in Figure 5.3 as described in Table 5.3 and the pre-defined value equal to 26. The abnormal coefficient, recognition results of system output and human recognition results are shown as Table 5.4. From the vision of observers, the motions in array 6 are abnormal motions, but the system regards they are normal. It is the wrong recognition. Even though the ratio of correctly distinguishing a single image from the image database is about 70%, the recognition of a motion array can be up to 87.5% in this example. It means that 87.5 percent it will trigger an alarm correctly.

Table 5.3 Security level of images in database

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>3</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>9</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>8</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
Table 5.4 Recognition of image array
(0: Normal, 1: Abnormal)

<table>
<thead>
<tr>
<th>Array</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abnormal coefficient</td>
<td>15</td>
<td>40</td>
<td>20</td>
<td>28</td>
<td>16</td>
<td>24</td>
<td>16</td>
<td>45</td>
</tr>
<tr>
<td>System output</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Human recognition</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

5.4 Embedded Design Method

Embedded controllers for reactive real-time applications are implemented as mixed software-hardware (S/W) systems. These controllers utilize micro-processors and digital signal processors but are neither used nor perceived as computers. Generally, software is used for features and flexibility, while hardware is used for performance.

To build an embedded system, first decide the purpose which is the basis for the behavior mode and system structures of the embedded system (Tiwari et. al, 2003), according to which the hardware devices and the software development tool are selected. Figure 5.8 shows the structure diagram of the embedded system development tool. Figure 5.9 shows system design and evaluation. The framework of the embedded system development tool includes human-machine interface, system programs, simulation programs and collection of modules (CPU, I/O and function modules). Outputs of the entire system are object files and hardware specification files. Users create hardware according to the hardware specification file, and then combo it into ROM to obtain the required embedded system. Figure 5.10 shows their corresponding flow chart. From the both figures, after the tasks are decided, the developed programs are complied to be object files, and then based on the selected hardware components, the programs are simulated. If the results are satisfactory, execute programs combining software and hardware. Finally, the interface shows the execution results of the software and some embedded modules.
Figure 5.8 Structure diagram of the embedded system development tool.

Figure 5.9 System design and evaluation

Figure 5.10 Flow chart of embedded systems
In the design flow, high level language translation provides to write the hardware and software specifications in a high level language that can be directly translated into co-design procedure. System level hardware-software co-simulation is a way to give feedback on different design choices. These design choices include HW-SW partitioning, CPU selection, and scheduler selection. Fast timed co-simulation (up to millions of clock cycles per second on a workstation) is possible thanks to the software synthesis and performance estimation techniques described below. Design partitioning means making system-level design decisions such as H/S partitioning, target architecture selection, and scheduler selection. These decisions are based heavily on design experience and are very difficult to automate. Therefore an environment is provided to quickly evaluate any such decision through various feedback mechanisms from either formal verification or system co-simulation. An application-specific OS, consisting of a scheduler and I/O drivers, is generated for each partitioned design. Interfaces between different implementation domains (hardware-software) are also automatically synthesized. These interfaces come in the form of cooperating circuits and software procedures (I/O drivers) embedded in the synthesized implementation. Communication can be through I/O ports available on the micro-controller, or general memory mapped I/O.

5.5 System Implementation of H/S Co-design

5.5.1 Flow of Hardware Design
In this research an efficient security system is designed to be installed into the automotive in order to screen dangerous invaders for safety problems automatically and intelligently. The digital cameras can be used to monitor spaces for the presence of individuals, whose digital images are compared to the image data in database, and trigger an alarm if the individual’s motions are abnormal.

The camera is fixed into the rear mirror and monitors the surrounding of the automotive, getting the real time image data and transfers it through wireless communication, from the sending module to the receiving module of Ni3. And then the image is transferred to the FPGA board, which is put in the cabin. At the same time the feature vectors of image data are extracted using ICA, clustered. Moreover, an array of feature vectors extracted using ICA for comparing and a pattern matrix has been already stored in RAM on the FPGA board. Also, because the pattern matrix \( W \) has already been stored in RAM on the FPGA board, the feature vectors of shot images can be extracted. From comparing the feature vectors of shot images and the centroid of cluster in the RAM, it is possible to respectively select the most similar clusters the shot images belong to, and after that it can be decided that which images in this cluster are the most
similar to the shot images. Therefore the security level of these shot images can be achieved and it can be decided whether these motions are normal or not. The system structure is shown as Figure 5.11.

A digital camera embedded on the PCB board is used to capture the images of individuals. The camera controller MCU is implemented by Xilinx low price FPGA Sparan-2 with IP core of MicroBlaze, the design flow can be referred to PowerPC introduced following. Furthermore the wireless sensor module of Ni3 consists of PIC MPU with A/D converter, analog/digital signal processing circuits and an RF transceiver. The PIC MPU with nonvolatile memory is used in receiving and sending modules. Only a minimal set of instructions are developed in the MPU in the digital processing without operation system, assuring the lower power consumption and smaller size of device. In the description below, the relative hardware devices are introduced first, and then the co-design of hardware and software are discussed to implement the system.

In the system, the communication tasks are completed by the wireless and network modules. Figure 5.12 shows the functional block diagram of the wireless module and the network module (I. Urushibara, et. al, 2004). As an input/output interface, the wireless module not only converts the analogue signals from the connected sensors to digital signals, but also has a function of an RF transceiver for radio communication with the network. On the other hand, the microcontroller embedded in the wireless module controls wireless communication. The module uses low-power radio standard, which is defined by Japan’s Radio Law, at the frequency of 303.825MHz. Depending on the surrounding environment, the possible distance of its data transmission is 15 meters. The maximum speed of wireless data transmission is 19.2 Kbps. As it is known that the changing speed of human activity is not fast, at least, the transmission speed of 19.2 Kbps is enough.
In the wireless module there is a MCU with the type of PIC16LF877A, for which users can download their programs freely to the MCU for special purposes. Besides the MCU, there is a SRAM with the size of 16KB on the wireless module for the purpose of storing large of data. The I/O ports of the module include five A/D converter input and three digital I/O ports, which means only one base board can connect to several sensors and control multiple signals. The module can also be applied for several communication types: 1 to N, multi-hop and ad-hop. Because of its low-cost power design, small batteries such as CR2032 can provide enough power. Moreover, the network module has the functions of serial and USB communication, which is optional. For the reason of the advantages of Ni3, it is selected to compose the data transfer system.

Except for the communication function, the other functions are basically implemented on Memec design development board of type Virtex II P7-672 with P160 communications module. The Virtex-II Pro contains platform FPGAs for designs that are based on IP cores and customized modules. The family incorporates multi-gigabit transceivers and PowerPC CPU blocks in Virtex-II Pro Series FPGA architecture. It empowers complete solutions for telecommunication, wireless, networking, video, and DSP applications. Virtex-II Pro architecture is optimized for high performance designs in a wide range of densities. Combining a wide variety of flexible features and IP cores, the Virtex-II Pro family enhances programmable logic design capabilities and is a powerful alternative to mask-programmed gate arrays.

The IP core of IBM PowerPC and the communications module are used. The module provides data interface to exchange data with outside. This system is designed by mainly taking advantages of PowerPC core. It is a 32-bit implementation of the
PowerPC™ embedded environment architecture. The PowerPC architecture not only provides a software model that ensures compatibility between implementations of the PowerPC family of microprocessors, but also defines parameters that guarantee compatible processor implementations at the application-program level, allowing broad flexibility in the development of derivative PowerPC implementations that meet specific market requirements.

The hardware system is developed using a serial of IP cores and bases on bus structure. A PowerPC is connected with two 16K Processor Local Bus (PLB) memories over the PLB bus. A UARTLite, serial communication IP core, and an Ethernet 10/100 MAC are connected to On-chip Peripheral Bus (OPB) providing the interface of data transfer between this chip and other parts of equipment and image database. A 32M OPB memory controller is connected to the OPB bus storing the data and instructions of compiled source code. This system also contains an interrupt controller to expand the number of interrupt inputs to PowerPC and request service. A timer serves to coordinate and handle the event of time expiring from UARTLite and MAC to complete the required system functions. A PLB to OPB bridge is used to connect the two different bus interface, refer to (2003c) for details. The hardware structure is shown in Figure 5.13 in detail.

Based on the hardware on the employed FPGA and multiple tools, users are allowed to design the complete embedded system whose design consists of the creation of the hardware and software components of the embedded processor system. Figure 5.14 illustrates the hardware and software co-design framework.
In Figure 5.14, the hardware platform is defined by the MHS (Microprocessor Hardware Specification) file. The hardware platform consists of one or more processors and peripherals connected to the processor buses. Several useful peripherals are usually supplied by Xilinx, along with the EST tools. The XPS tool provides graphical means to create the MHS file. The MHS file defines the system architecture, peripherals and embedded processors. The MHS file also defines the connectivity of the system, the address map of each peripheral in the system and configurable options for each peripheral. Multiple processor instances connected to one or more peripherals through one or more buses and bridges can also be specified in the MHS.

5.5.2 Flow of Software Design

The software platform is defined by the MSS (Microprocessor Software Specification) file. The MSS file defines driver and library customization parameters for peripherals, processor customization parameters, standard input/output devices, interrupt handler routines, and other related software features. The software application is the code that runs on the hardware and software platforms. The source code for the application is written in a high level language such as C or C++, or in assembly language. Once the source files are created, they are compiled and linked to generate executable files in the ELF (Executable and Link Format) format. GNU compiler tools for PowerPC and MicroBlaze are used by default but other compiler tools that support the specific processors used in the hardware platform may be used as well. XMD and the GNU debugger (GDB) are used together to debug the software application. XMD provides an instruction set simulator, and optionally connects to a working hardware platform to allow GDB to run the user application.
The hardware system can be synthesized and verified by EDA tools, and therefore it needs more focusing on the software design for this system. After simulate the main functions of software system on the Matlab platform, C++ source code is developed and compiled using GNU compile to verify the stability and test the relative time consumption. C++ source code can be compiled and run on PowerPC directly. A serial of development tools are used which are efficient to build the hardware and software systems. The frame of hardware and software co-design includes the development of hardware platform of IP cores and software platform of C++ source code with Xilinx EDK tools.

In this system, the memory usage is also optimized. The PLB memory is integrated in FPGA with high speed and high cost. The OPB memory is common the SDRAM, it is slow but cheap. The PLB memory is used to store the data and instructions frequently used and leave the amount of image data into OPB memory. The dynamic memory allocation technology is used in the procedure of system memory management in the OPB memory. The memory of big amount is allocated in the heap region and can be released and reallocated for further use. These lower the cost of entire system and increase efficiency greatly. There is no RTOS in this system from the consideration of cost and complexity. Interrupt mode is used to improve the system response time and compensate the efficiency lost from the lack of threads in RTOS. For example the receiving of shot image data will trigger an interrupt to PowerPC and get the CPU time for processing. Also the interrupt processing routine will notify the database image processing subroutines after data receiving. This method can satisfy the real time requirements of the system.

5.6 Summary

This chapter illustrates a novel application of Independent Component Analysis to the automotive security system. First the image database is defined to represent abnormal motions of people around the door of automotive according to the opinions of observers. The array of feature vectors and the pattern matrix of images in the database extracted using ICA are stored in RAM on the FPGA board, therefore the image database is seamlessly integrated to the real time system. For large image database the image data are clustered to improve the efficiency and precision. A novel similarity calculation method is developed to recognize the most similar image in the certain cluster according to the extracted feature vectors. By comparing the feature vectors of shot image and the cluster centroid, the most similar image to database image can be achieved from the corresponding cluster. The security level of a person can be calculated and any appearance deemed with threat can be set to trigger an alarm. The system is simulated on MATLAB and Visual C++ and implemented into FPGA for real time monitor.
Chapter 6 Conclusions

In the recent years, more and more equipments in automotives are changing from mechanical systems to electronic systems. Embedded system is a core of vehicle electronic systems because of its flexibility and versatility. The electronics revolution has influenced almost every aspect of automotive design including the powertrain, fuel combustion, crash protection and the creation of a comfortable cabin and nearly wireless environment. It is necessary to pay more attention to the fields of environments, safety and security, which are the most significant and challenge field of automotive embedded system design.

The section of engine control system proposes a novel event detection and location mechanism, extends the traditional automaton and develops a new simulation algorithm of hybrid system. The simulation of hybrid system combines the integration of continuous parts and the processing of discontinuous transitions. In the proposed approach the sign of the event function is monitored and an event in the span of a step is searched. When the sign of multiplication of event function changes from positive to negative or from negative to positive, an event happens and the system trajectories cross the switching surface. A first-in-first-out stack is used to store the calculated approximation of variable states. The switching situations existing on the sliding surface are analyzed, the well-defined control semantics are developed for managing the behavior discontinuities in the case of state transition as event processing. The calculation algorithm that can seamlessly combine continuous behavior generation with discrete mode switching is designed. The intake manifold model of engine with turbo charger is built and the model equations are solved using the proposed simulation method for model based predictive control in real time. This method can solve the model under entire region of input throttle angles. Furthermore, the stability is greatly increased to restrict the error less than 1% and the calculation time is greatly reduced of eight times for real time control system. This model is verified using Matlab and C++ and can be embedded into engine control system to calculate the air flow mass for real time predictive control.

The section of automotive safety of ABS explores a novel method of building a virtual vehicle system, integrating pure software simulation and hardware in loop (HIL) simulation which is constructed based on distributed network of CAN and TCP/IP. After that a prototype of ABS controller based on fuzzy control theory is developed. First, the dynamic
vehicle model is developed in the platform of software simulation, used to design and verify the response of ABS control logic; Second, in HIL simulation the model equations are converted to a stand-alone program of C++ on the main computer and integrated with ABS controller, real vehicle devices and components, which generates a real vehicle environment based on CAN bus to verify the control performance of ABS in real time. Database, graphic and data processing system are integrated by cross-platform programming technology on the assistant computer connected by Ethernet. An integrated user-friendly interface including vehicle parameters database editors, configurations and visualization tools is also developed to interact with the core components. MATLAB, database and EXCEL are integrated into this system through Visual C++ programming platform on assistant computer to make the system easy to users. The prototype of ABS controller is designed based on the fuzzy control theory. The analysis and validation of developed control logic for ABS can be performed successfully on this virtual platform.

The section of automotive security explores the development of a real time intelligent security system using the technology of pattern recognition based on independent component analysis (ICA) and a novel matching method as a reaction to the perception of auto theft. The images representing abnormal motions of people around the door of automotive are shot and stored into image database. They are used for motion recognition by matching the real time image caught by the micro camera. The feature vectors of images are extracted using ICA and organized into categories by fuzzy c-mean cluster algorithm in order to improve the precision and decrease time consumption. An array of motion images of people are caught by a micro digital camera and transferred through wireless networks to FPGA board. The feature vectors of shot images are extracted using pattern matrix on IP core of PowerPC. By comparing the feature vectors of shot image and the cluster centroid of database images, it is possible to obtain the cluster to which the shot images belong to respectively. After that the most similar image can be achieved from the corresponding cluster. The security level of a person can be calculated and any appearance of a person deemed threatening can be set to trigger an alarm. The system of hardware and software co-design is implemented on Xilinx FPGA with the performance of high efficiency, low power consumption and easy integration with other devices.
Acknowledgements

No man is an island. And no research is done in a vacuum. My life and research at Graduate School of Information, Production and Systems, Waseda University have been influenced by, and have benefited from, the involvement with a number of extraordinary advisors, teachers, colleagues, and friends.

Foremost, I would like to express my deep gratitude to my thesis supervisor, Prof. Ogai. He has been a good advisor and supporter, but most importantly an example of a true scholar. I admire and owe him for his broad knowledge, connectionist approach, and deep insight as well as for his trust and perseverance. I also owe him for managing a lab where students can explore new directions and for creating opportunities for me to learn and grow as a researcher.

Further, I particularly thank Prof. Inoue in Waseda University and Dr. Ohata, the manager in Toyota Motor Cooperation. The work of engine intake control in this dissertation is jointed with both of them. Prof. Inoue offers helpful comments and assistance throughout. This work is catalyzed by Mr. Ohata’s eagerness to listen, quick mind, and profound problem-solving ability. I also thank Prof. Yamauchi in Waseda University. The work of security system in my dissertation is cooperated with him. He provides me with a helpful hardware platform.

While at Graduate School of IPS, Waseda University, I have had the good fortune to be taught by the best minds in systems, control theory and embedded software and hardware co-design, including Professor Yamauchi, Professor Lee, Professor Watanabe, and so on.

And I will express my special appreciation to Professor Ogai, Professor Inoue, Professor Yamauchi and Professor Lee for their work of reviewing my thesis. Further, I am indebted to the lab students who generally put up with me and helped me get things done throughout the years.

For part of this work, I was supported by Toyota Motor Cooperation, which is greatly appreciated. This work is also supported by Kitakyushu Foundation for Industry and Science. Their assistance is gratefully acknowledged.

Finally, I would like to thank the friends and family who have always been there, despite the ravages of time and distance. Thanks Qian Tian. She always encourages and helps me to overcome the handicap in my research and life, and also gives me a big help to complete my thesis. Thanks Mom, and Dad. Without you this victory would be hollow, if
at all possible.

To everyone, I hope our paths continue to cross through the years and wish you happiness and good fortune in all that you do.
References


Publication List

International Journals:

Proceedings of International Conferences

Proceedings of Domestic Conferences
年計測自動制御学会制システムインテグレーション講演会, 熊本電波工業高等専門学校, 平成17年12月16日, pp. 867-868.