Learning and Input Selection of Human Strategy in Controlling a Single Wheel Robot

by

Wai-Kuen Yu

A dissertation submitted in partial

fulfillment of the requirements for the degree of

Master of Philosophy

in the

Department of Mechanical and Automation Engineering

of

The Chinese University of Hong Kong

Shatin, N.T.,

Hong Kong, China

Learning and Input Selection of Human Strategy in Controlling a Single Wheel Robot

摘 要

在近幾年裏，一種被稱爲"Gyrover"的單輪機器人正在成爲熱門的機器人研究專案．Gyrover 是一種旋轉平衡類型的機器人，它最初由卡內基梅隆大學開發．這種機器人有許多潛在的用途，例如，基於它可以在崎嶇不平或軟質的地面上高速賓士的特點，它可廣泛地應用於運輸，探險及拯救等作業．

由於這種機器人的高耦合性，非完整性和低驅動等特性，使得傳統的控制方法應用在其上會產生一連串的問題．另外，考慮到人具有能夠自如地控制複雜的機器人的技能，這啓發我們得到首先研究人的控制技能，然後再將它傳授給該機器人的想法．我們發現，從自動機器人到人機協調的廣泛領域中，模型化人類控制技能(HCS)的方法被大量使用，借助於 HCS 方法，本論文從中提取了適用於 Gyrover 的人的控制技能策略．

在應用模型化方法的過程中，爲提取人的控制技能，模型輸入的選取是相當重要的一步．由於存在很多的輸入變數，而且互相關性差的輸入變數會影響 HCS 模型的效能，所以應該選取一些能夠與動態 HCS 準確吻合的變數．爲此，我們發展了兩套應用在不同情況下評價方法，以幫助選取適當的變數．

在第一種評價方法中，我們首先從機器人的動態模型中選取相關的狀態變數，作爲 HCS 模型的輸入，然後，我們定義了一種相對於操作者控制輸入所產生的每一狀態變數敏感性的量度方法，以確認狀態變數的重要性．

對第二種評價方法，我們引用了一種稱爲因數分析的統計方法來選取合適的狀態變數．利用因數分析，變數將被分爲不同的組，在同一組內的變數互相關性強，而不同組間的變數的互相關性弱，這便於我們選取與人的控制策略有強相關性的一組作爲輸入變數．這種評價方法的優點是無須先對系統的動態模型有所認識，而且能夠推廣到任何一種未知模型的系統上，作爲分析和選取輸入變數的手段．

最後，我們分別應用兩種評價方法演示了：在已學習了人的控制輸入模型的前提下，這種機器人幾種運動方式的自動控制情況．演示表明，本論文對模型化人的控制技能，並應用于該機器人的動態穩定性控制是十分有意義的．

Abstract

A single wheel, gyroscopically stabilized robot, called Gyrover, is a novel concept of mobile robot which provides dynamic stability for rapid locomotion. It is a sharp-edged wheel actuated by a spinning flywheel for steering and a drive motor for propulsion. The spinning flywheel acts as a gyroscope to stabilize the robot. This configuration conveys significant advantages over mult-wheel, statically stable vehicles, including good dynamic stability and insensitivity to attitude disturbances; high maneuverability; low rolling resistance; ability to recover from falls and amphibious capability. Thus, successful development of autonomous control of such kind of robotic system will expand the range of robotic applications.

It is complicated and difficult to control the robot by classical control method because of its highly coupled, nonholonomic and underactuated nature. On the other hand, humans are capable of mastering complex, skill control to the robot. Taking advantage of human skill in teleoperation control of the robot, the goal of this thesis is to study the learning and transferring human strategy in controlling the robot. We apply the methodology in modeling the human control strategy (HCS) in this thesis, developed by Xu and Nechyba, to abstract human skill in driving on the robot. Then we develop a human-based controller for controlling such a dynamically stable but statically unstable robot.

In the modeling, the selection of model input plays an important role for extracting human skills. There are many parameters and variables in input and output space. It is impossible and inaccurate to include all variables in the modeling, therefore, selecting a set of variables that contributes significantly to the HCS is one of the most important tasks in the modeling. To this end, we develop two eval-

uation methods to select suitable input variables to handle the selection problem in different situations. In the first evaluation method, we first select the relevant state variables from the mathematical model of the robot as the HCS model input. Then, we define a measure of sensitivity of each state variables with respect to the operator's control input to validate the importance of model input. In the second evaluation method, we present a statistical method, *factor analysis* , to select the suitable state variables. By using factor analysis, the variables can be classified into different groups. Intra-group variables have higher correlations among themselves while inter-group variables have lower correlation among themselves. Then, we select the groups of variables which have high correlation with the human strategies as the input variables. The advantage of this evaluation method is that prior knowledge of the dynamic model of the system is unnecessary. By this feature, this method can be further applied to any systems without any model for analyzing and selecting input variables. Finally, we verified the HCS models through simulation, and experimentally implemented the human-based controller for controlling the "tilt-up" motion of the robot. We demonstrated that the robot can be automatically recovered from falling.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Prof. Yangsheng Xu for his patience and intellectual guidance on all aspects of the work.

I am indebted to Kwok-Wai Au for his invaluable comments and suggestions in my work. I would like to give special thanks to Prof. Michael C. Nechyba and Wai-Keung Fung for many helpful discussions on several issues. Thanks are also given to Prof. Jun Wang, Prof Jie Huang, Prof. Yunhui Liu, Jinyang Song, Ka-keung Lee for their helpful discussions during my research. Also, I would like to thank Winston Sun, Lo-Wai Sun, Chi-tin Yang, Wai-hung Tang, Yantao Shen , Yau-fai Kwok, Ming-ho Chan and Kin-fung Lei for their helps in these two years.

Finally, I would like to give my sincere appreciation to my parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Robot Concept

In this thesis, we concern about a robot called Gyrover which was originally developed at Carnegie Mellon University and has been continuously studied here in Hong Kong. Gyrover is a novel, single-wheel robot that is stabilized and steered by means of an internal, mechanical gyroscope. Figure 1.1 shows the latest version, the third prototype of Gyrover called Gyrover III. The robot was designed with equipping numerous inertial sensors and wireless communication. It includes a radio system for remotely control, and an on-board computer and sensors to permit data-acquisition and remotely control. It is built with a light-weight bicycle tire and a set of transparent domes so that the entire system is enclosed and thus, protected within the wheel.

This configuration conveys significant advantages over multi-wheel, statically stable vehicles, including

- good dynamic stability and insensitivity to attitude disturbances;
- high maneuverability;
- low rolling resistance;
- ability to recover from falls;

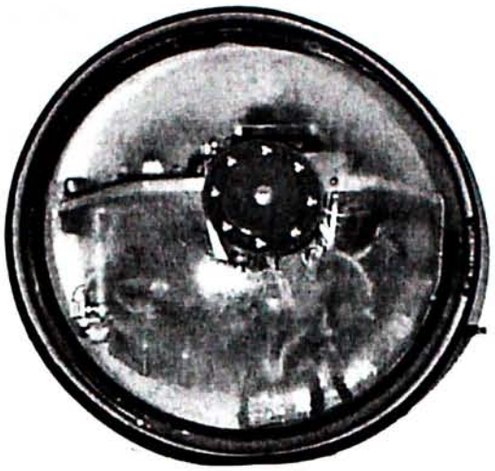

Figure 1.1: **The third prototype of Gyrover**

- amphibious capability.

The robot concept is based on the principle of gyroscopic precession as exhibited in the stability of a rolling wheel. Because of its angular momentum, a spinning wheel tends to precess at right angles to an applied torque, according to the fundamental equation of gyroscopic precession:

$$T = J \cdot \omega \times \Omega \tag{1.1}$$

where ω is the angular speed of the wheel, Ω is the wheel's precession rate, normal to the spin axis, J is the wheel polar moment of inertia about the spin axis, and T is the applied torque, normal to the spin and precession axes. Therefore, when a rolling wheel leans to one side, rather than just fall over, the gravitationally induced torque causes the wheel to precess so that it turns in the direction that it is leaning. the robot supplements this basic concept with the addition of an internal gyroscope — the spinning flywheel — nominally aligned with the wheel and spinning in the direction of forward motion. The flywheel's angular momentum produces lateral stability when the wheel is stopped or moving slowly. Figure 1.2 illustrates the concept of gyroscopic precession.

Potential applications of the robot are numerous. As it can travel on both land and water, it may find amphibious use on beach or swampy area, for general transportation, exploration, rescue or recreation. Similarly, with appropriate tread, it

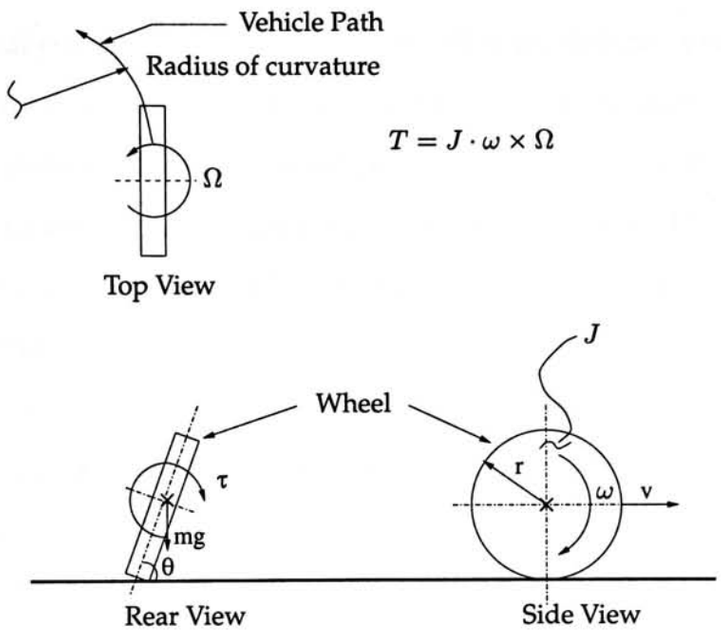

Figure 1.2: **Principle of gyroscopic precession.**

should travel well over soft snow with good traction and minimal rolling resistance. As a surveillance robot, the robot could use its slim profile to pass through doorways and narrow passages, and its ability to turn in place to maneuver in tight quarters. Another potential application is as a high-speed lunar vehicle, where the absence of aerodynamic disturbances and low gravity would permit efficient, high-speed mobility.

1.2 Motivations

The robot control is challenging due to the special characteristics of the robot[2]. First, it is a highly coupled dynamic system between the wheel and the flywheel. Second, it is subject to two nonholonomic constraints due to the rolling constraints and underactuation in the rolling direction. Third, it is inherently unstable in the lateral direction. Preliminary studies in the research on the modeling and control of the robot has been conducted. However, the model and the controller in [2] rely on the assumption that the robot rolls without slipping. Thus, some kinds of robot motions cannot be analyzed by using this model, such as "tilt-up" or sliding motions.

In the practical point of view, it is also difficult to model the system precisely because it largely depends on various unmodeled parameters, such as friction. These parameters are relatively more important in the dynamically stabilized system discussed here than a conventional static or quasic-static system, like a 4-wheel vehicle. Moreover, the existence of sliding motion of the robot makes it more difficult to obtain a complete model.

On the other hand, humans are capable of mastering complex, control skills for the robot. A human operator can drive the single wheel fluently after learning for a while. Thus, we naturally come up the idea of learning human control strategy (HCS) to abstract the operator 's driving skill and develop a human based controller for controlling the robot.

We can train different HCS models for carrying out different tasks. Thus, modeling HCS has potential impact in a number of applications ranging from autonomous control and teleoperation to human-robot coordination and human-machine system simulation. For example, the Intelligent Vehicle Highway System (IVHS), currently being developed through massive initiatives in the United States, Europe, and Japan[30], envisions automating much of the driving on our highways. The required automated vehicles will need significant intelligence to interact safely with variable road conditions and other traffic. Modeling human intelligence offers one way of building up the necessary skills for this type of intelligent machine.

Although modeling HCS has many potential applications, it introduces a challenging problems in implementation: input selection problem - Human control data are acquired by sensors for modeling HCS. There are numerous sensor variables and their corresponding derivatives available from the systems,e.g. Euler angles (roll-pitch-yaw) of the robot and their corresponding angular velocities and angular accelerations. What variables should we select as model input?

In this thesis, our goal is to design a human-based controller for the robot. For effective modeling HCS, the robot learns the correct behaviours by observing suitable variables from the human control data. In our approach, we apply cascade neural network to model human skill through learning experimentally by suitable input selection.

1.3 Related Work

There are some researches on the single wheel robot because it is a novel concept, only a few studies in modeling and control of the robot have been carried out. Xu and Au [2][5][10][11] have done on modeling and control of Gyrover. They developed a mathematical model for Gyrover and designed a straight line tracking controller. Also, Xu and Sun [12] has worked on stabilization of Gyrover on an inclined plane by a back-stepping controller. These studies solely rely on analytical approach and depend on the assumption that the wheel rolls without slipping. some motions are invalid to this assumption. On the other hand, HCS controller has potential to be developed as a controller in this situations because modeling HCS is a general methodology to abstract human skill and do not depend on this assumption. It can be applied on different cases provided that the operator is able to control Gyrover in these cases.

Early research in modeling human skills was based on control theory paradigm, which attempted to model in the loop as a simple feedback control system [41]. By that approach, human was often modeled as a simple time delay in the overall human machine system. In recent works, different approaches have been done to learn more advanced human skills. Fuzzy control is one type of approaches. In fuzzy control schemes[36][37], human experts are asked to specify "if-then" control rules with fuzzy linguistic variables. And it has been demonstrated for automobile steering and ships helmsmen [42]. Although fuzzy system is well suited to apply for control tasks with few inputs and outputs, they do not scale well to the system with high dimensional input space and output space. It means that the "if-then" rule-based is too complex to handle the input-output relationship. Asade and Liu derived control rules from human input pattern and corresponding output actions to a deburring robot[13][14]. They used Lipchitz 's condition to verify the consistency of the human control data. But they only implemented the method on the statically stable system with few input variables. Yang, Xu and Chen[5] applied Hidden Markov Models (HMM) to open-loop skill learning in the telel-operation control of a space system. Nechyba and Xu [1][6][7] applied machine

learning techniques and statistical analysis towards abstracting models of human control strategy. The learning architecture is based on flexible cascade neural network with node-decoupled extended Kalman filtering. They also validated the models by similarity measure, which measure the level of similarity between mult-dimensional, stochastic trajectories .

The researches so far has not addressed the input selection problem as we mentioned in the above section. Most recent work in learning from human data dealt with either action skills, quasi-static skills, or high-level abstraction of human skill (e.g., assembly). People usually interested in learning methodology rather than input selection. Since most intelligent control methods applied on the problems that already had a solution by classical control before. e.g. neural network applied on control a robot manipulator [39] and fuzzy applied on control an inverted pendulum[40]. Therefore, the important variables for controlling the systems are already well-known. People can directly apply the intelligent methods with already well-known input variables. However, in our case, we are still developing the control system of the robot and we only partially understand the mechanism of the robot , and therefore, we need to find the variables for model input.

1.4 Overview

In our project, we focus on development of selection methods to analyze and select suitable state variables for HCS model input. We experimentally implement HCS model with input selection method on the robot. The thesis is organized as follows:

- Chapter 2: We will illustrate the general frame work of the single wheel robot. We will first describe the mathematical model of the single wheel robot which has been developed by Xu, Au and Brown [2][9][10]. Then, we will introduce the system development of the third prototype of the robot for automatic control. We will present hardware component and the software design of the robot.

- Chapter 3: In this chapter, we describe the methodology to abstract human control strategy(HCS). Then, we implement the methodology in controlling the robot. We first record the operator's control skills and the corresponding states of the robot when he drives the robot. Second, we use these data to train HCS model by using the flexible cascade neural network learning architecture with node-decoupled extend Kalman filtering. Third, we transfer the HCS model in controlling the single wheel robot. We experimentally demonstrate that the robot can be automatically controlled by the Human-based controller.

- Chapter 4: we develop two methods to select suitable input variables to handle the selection problem in different situations. In the first evaluation method, we first select the relevant state variables from the mathematical model of the robot as the HCS model input. Then, we define a measure of sensitivity of each state variables with respect to the operator's control input to validate the importance of model input. In the second evaluation method, we present a statistical method, *factor analysis* , to select the suitable state variables. By using factor analysis, the variables can be classified into different groups. Then, we select the groups of variables which have higher correlation with the human strategies as the input variables.

Chapter 2

Single Wheel Robot

The single wheel robot can be considered as a single wheel actuated by a spinning flywheel attached to a two-link manipulator at the wheel bearing and drive motor [2]. The robot uses the spinning flywheel, as a gyroscope, to stabilize itself. The single robot have several advantages over multi-wheel, statically stable vehicle. The advantages include good dynamic stability and insensitivity to attitude disturbances, high maneuverability, low rolling resistance and ability to recover from fall.

In this chapter, we will first describe the mathematical model of the single wheel robot which has been developed by Xu, Au and Brown [2][9][10]. Then, we present the system development of the robot for automatic control including hardware component and the software design of the robot.

2.1 Mathematical Model

In this section, we present the simplified model of the single wheel robot. For details, see Appendix A or [2] for the description of the mathematical model of the robot. The following description in this section is the work done by Xu et. al. [2][9][10].

2.1.1 Coordinate Frame

Figure 2.1 illustrates the general coordinates$(X_c, Y_c, \alpha, \beta, \gamma)$ for the single wheel robot. Four coordinate frames are defined as follow: (1) the inertia frame $\sum_O$, (2) the body coordinate frame $\sum_B \{x_B, y_B, z_B\}$, (3) the coordinate frame of internal mechanism $\sum_C \{x_c, y_c, z_c\}$, whose center is located at point D, and whose z-axis is always parallel to z_B, and (4) the flywheel coordinates frame $\sum_E \{x_a, y_a, z_a\}$, whose center is located at the center of the flywheel, and whose z-axis represents the axis of rotation of the flywheel. Link l_1 is rotated about the z_B-axis by a swing angle, θ. The swing angle is zero when link l_1 is parallel to x_B axis. The flywheel is tilted about the y_c-axis by the tilt angle , $\beta_a \in (0, \pi)$. Note that y_a is always parallel to y_c. Therefore, the configuration of the single wheel robot can be described by seven generalized coordinated $(X_c, Y_c, \alpha, \beta, \gamma, \theta, \beta_a)$. The definition of model variables is shown in table 2.1.

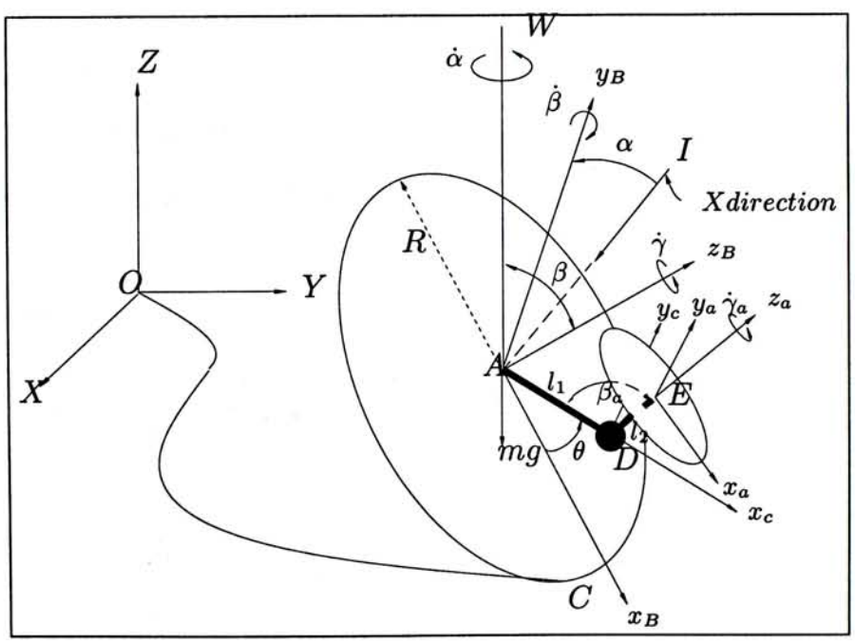

Figure 2.1: **Definition of coordinate frames and system variables**

Table 2.1: **Variables Definition**

α, α_a	Precession angles of the wheel and for the flywheel, respectively, measured about the vertical axis
β	Lean angles of the wheel
β_a	Tilt angle between the link l_1 and z_a-axis of the flywheel
γ, γ_a	Spin angles of the wheel and the flywheel, respectively
θ	Angle between link l_1 and x_B-axis of the wheel
m_w, m_i, m_f	Mass of the wheel, mass of the internal mechanism and mass of the flywheel respectively
m	Total mass of the robot
R, r	Radius of the wheel and the flywheel respectively
$I_{xxw}, I_{yyw}, I_{zzw}$	Moment of inertia of the wheel about x, y and z axes
$I_{xxf}, I_{yyf}, I_{zzf}$	Moment of inertia of the flywheel about x, y and z axes
μ_s, μ_g	Friction coefficient in yaw and pitch directions, respectively
u_0, u_1	Drive torque of the drive motor and tilt torque of the tilt motor, respectively

2.1.2 Equations of Motion

The equation of motion can be derived by calculating the Lagrangian $L = T - P$ of the system, where T and P are the kinetic energy and potential energy of the system respectively. The system can be divided into three parts: 1) wheel, 2) internal mechanism and 3) spinning flywheel.

The spinning rate of the flywheel always keeps constant because it is reluctance to change the spinning rate due to the large moment of inertia, thus torque

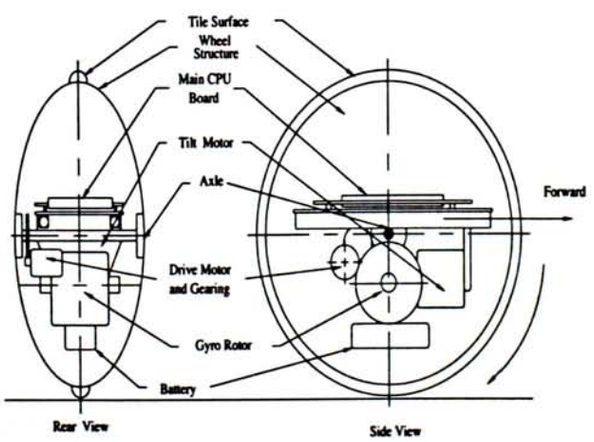

Figure 2.2: **The basic configuration of Gyrover**

acting on the spin motor can be approximately constant. Thus, we only concern on two generalized force acting the single wheel robot. One is drive torque (u_0) and the other is the tilt torque (u_1). Consequently, using the constrained Lagrangian method, the dynamic equation of entire system is given by,

$$M(q)\ddot{q} + N(q,\dot{q}) = A^T\lambda + Bu; \tag{2.1}$$

Where $M(q) \in R^{7\times7}$ and $N(q,\dot{q}) \in R^{7\times1}$ are the inertia matrix and nonlinear terms respectively.

$$A(q) = \begin{bmatrix} 1 & 0 & -Rc\alpha c\beta & Rs\alpha s\beta & -Rc\alpha & 0 & 0 \\ 0 & 1 & -Rc\beta s\alpha & -Rc\alpha s\beta & -Rs\alpha & 0 & 0 \end{bmatrix} \tag{2.2}$$

$$q = \begin{bmatrix} X \\ Y \\ \alpha \\ \beta \\ \gamma \\ \beta_a \\ \theta \end{bmatrix}, \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ k_1 & 0 \\ 0 & 1 \\ k_2 & 0 \end{bmatrix}, u = \begin{bmatrix} u_0 \\ u_1 \end{bmatrix}$$

The nonholonomic constraints can be written as,

$$A(\dot{q})\dot{q} = 0. \tag{2.3}$$

It is noted that *the last two columns of matrix A are all zero* as the nonholonomic constraints only restrict the motion of the wheel, not the flywheel. The last two columns represent the motion variables of the flywheel. Moreover, *the matrix B only have three rows that are nonzero* since the input torques only drive the tilt angle of the flywheel (β_a) and the rotating angle of the wheel (γ), so that the fifth and the sixth rows of B are non-zero as they represent the tilting motion of the flywheel and the rotating motion of the wheel respectively. Furthermore, when the wheel rotates, the pendulum motion of internal mechanism is introduced, thus θ changes. Therefore, the drive torque of the wheel will also affect the pendulum motion of the internal mechanism (θ), so that the seventh row of the matrix B is not zero.

2.1.3 Model Simplification

Practically, l_1 and l_2 are assumed to be zero, thus mass center of flywheel and the internal mechanism are coincident with the center of the robot. Moreover, the pendulum motion of the internal mechanism is sufficiently small to be neglected, thus θ equal to zero. The spinning rate of the flywheel $\dot{\gamma}$ is assigned to be constant. Based on the previous derivation, the normal form of the mathematical model is

$$\bar{M}(q)\ddot{q} = \bar{F}(q,\dot{q}) + Bu \tag{2.4}$$

where $q = [\alpha, \beta, \gamma, \beta_a]^T$,

$$\bar{M} = \begin{bmatrix} \bar{M}_{11} & 0 & \bar{M}_{13} & 0 \\ 0 & I_{xxf} + I_{xxw} + mR^2 & 0 & I_{xxf} \\ \bar{M}_{13} & 0 & 2I_{xxw} + mR^2 & 0 \\ 0 & I_{xxf} & 0 & I_{xxf} \end{bmatrix},$$

$$\bar{F} = [\bar{F}_1, \bar{F}_2, \bar{F}_3, \bar{F}_4]^T,$$

$$B = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T, u = \begin{bmatrix} u_0 \\ u_1 \end{bmatrix}$$

$$\begin{aligned}
\bar{M}_{11} &= I_{xxf} + I_{xxw} + I_{xxw}C_\beta^2 + mR^2C_\beta^2 + I_{xxf}C_{\beta,\beta_a}^2 \\
\bar{M}_{13} &= 2I_{xxw}C_\beta + mR^2C_\beta \\
\bar{F}_1 &= (I_{xxw} + mR^2)S_{2\beta}\dot{\alpha}\dot{\beta} + I_{xxf}S_{2\beta\beta_a}\dot{\alpha}\dot{\beta} + I_{xxf}S_{2\beta\beta_a}\dot{\alpha}\dot{\beta}_a \\
&\quad +2I_{xxw}S_\beta\dot{\beta}\dot{\gamma} + 2I_{xxf}S_{\beta,\beta_a}\dot{\beta}\dot{\gamma}_a + 2I_{xf}S_{\beta,\beta_a}\dot{\beta}_a\dot{\gamma}_a - \mu_s\dot{\alpha} \\
\bar{F}_2 &= -gmRC_\beta - (I_{xxw} + mR^2)C_\beta S_\beta\dot{\alpha}^2 - I_{xxf}C_{\beta,\beta_a}S_{\beta,\beta_a}\dot{\alpha}^2 \\
&\quad -(2I_{xxw} + mR^2)S_\beta\dot{\alpha}\dot{\gamma} - 2I_{xxf}S_{\beta,\beta_a}\dot{\alpha}\dot{\gamma}_a \\
\bar{F}_3 &= 2(I_{xxw} + mR^2)S_\beta\dot{\alpha}\dot{\beta} \\
\bar{F}_4 &= -I_{xxf}C_{\beta,\beta_a}S_{\beta,\beta_a}\dot{\alpha}^2 - 2I_{xxf}S_{\beta,\beta_a}\dot{\alpha}\dot{\gamma}_a
\end{aligned}$$

where $\bar{M}(q) \in R^{4\times4}$ and $\bar{F}(q,\dot{q}) \in R^{4\times1}$ are the inertial matrix and nonlinear term of the single wheel robot respectively. The description of the notions shows in the

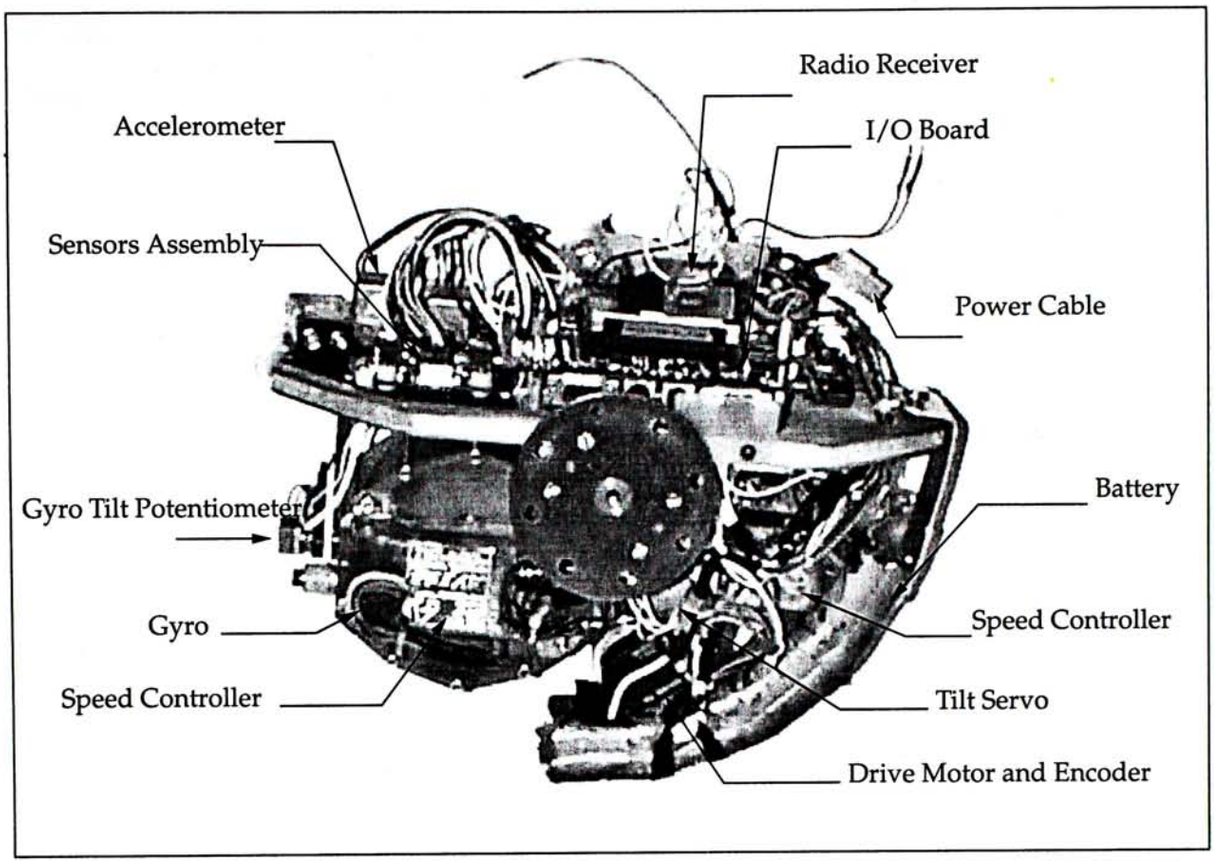

Figure 2.3: **Internal configuration of single wheel robot.**

table 2.1. For details, the appendix will describe the mathematical model of the Gyrover.

2.2 Hardware Descriptions

Compared with the previous two prototype of the robot, the third version was designed on a larger scale to equip numerous inertial sensors and a computer (486 PC) for data acquisition and control. The single wheel robot was built with a lightweight bicycle tire and a set of transparent domes so that the entire system is enclosed. The robot travels up to 10 mph and runs about 25 minutes per charge of its NiCad batteries. An internal gyroscope (high spinning flywheel) is installed and is maintained a constant angular velocity (approximately 1500 RPM) by servo controlled motor. Overall weight is about 7 kg.

In this section, the main components of the hardware will be described as

follows: actuators, sensors, communication subsystem and computer subsystem. Then, we will overview the structure of the robot's hardware.

2.2.1 Actuators

The robot consists of three actuators. They are

1. Drive motor u_0.- It drives the wheel and thus, moves the robot forward and backward.

2. Tilt motor u_1.- It tilts the angle of Gyro and thus, generates a torque, normal to the spin and precession axes.

3. Spin motor u_2.- It spins the flywheel and thus, increases the angular momentum of flywheel.

In our experiments, the speed of spin motor is always kept at constant since the operator usually controls the motors u_0, u_1 via two joystick of the radio transmitter (figure 2.4). operator can control the u_2 by pressing the other bottom on the radio transmitter. For convenience, we usually set maximum speed of flywheel.

2.2.2 Sensors

A number of on-board sensors have been installed on the the robot to provide information about the state of the machine to the on-board computer. They can measure :

- Gyro tilt angle
- Tilt servo current
- Drive motor current
- Drive motor position/speed
- Gyro speed

- Acceleration (three axes)
- Angular rate (three axes)
- Tilt angle (two axes)

All these signals, plus the control input from the radio transmitter, can be read by the on-board computer. The signals may be in the form of analog input quadrature (e.g. encoder) inputs; pulse frequency (e.g. Hall sensors); or pulse-width modulated signal (standard R/C signals).

The tilt potentiometer is to measure the tilt angle of the gyro. The two pulse encoders are to measure the spinning rate of flywheel and the single wheel. The two gyros and accelerometer are to detect the angular velocity of yaw, pitch, roll, and acceleration respectively. A 2-axis tilt sensor is developed and installed for direct measuring the leaning and pitch angle of the robot. The two current-sense resistors are to measure tilt servo current and drive motor current. These sensors and others electronic devices are essential for automatic control.

Some sensors such as tilt sensor can acquire accurate, low noisy data. Meanwhile, some sensors such as accelerometer acquire noisy data. Figure 2.7 shows some data acquired by sensors.

2.2.3 Communication Subsystem

The communication subsystem is composed of radio links which transmit data and commands between the robot and the ground station or the radio transmitter. The robot can be remotely controlled by these two remote consoles (Figure 2.4). For the radio transmitter, human operator remotely controls the robot by the two joysticks of the transmitter. He/she controls the drive speed and tilt angle of the robot via the transmitter. The ground station is to download the sensor data file from the on-board computer and start/terminate any program in the robot. For example, human operator controls the robot via the radio transmitter and then, the ground station downloads the sensor data file to the laptop so that we can analyze the real-time motion.

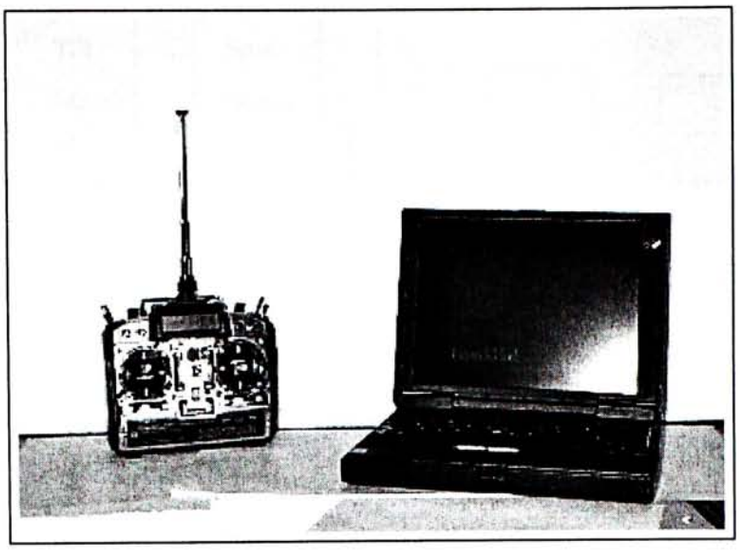

Figure 2.4: **Communication equipments: radio transmitter (left) and laptops with wireless Modem (right).**

2.2.4 Computer Subsystem

A 486 computer embedded on the robot to provides enough computation power for complicated task. The computer also responds for integrating other components of the robot and thus, the circuit board contains

- interface circuitry for the radio system and servos
- logic components to control power of the actuators
- interface for the on-board sensors

All sensors and actuators are connected to the on-board computer. Figure 2.5 shows the hardware architecture of the robot.

2.3 Software Descriptions

The complexity of the robotic system being developed requires a software architecture where different subsystems can run independently and in parallel, while able to exchange information an activate or inhibit each other. Therefore, in our robotic system, a real-time operating system, QNX, is used to run HCS-based controller

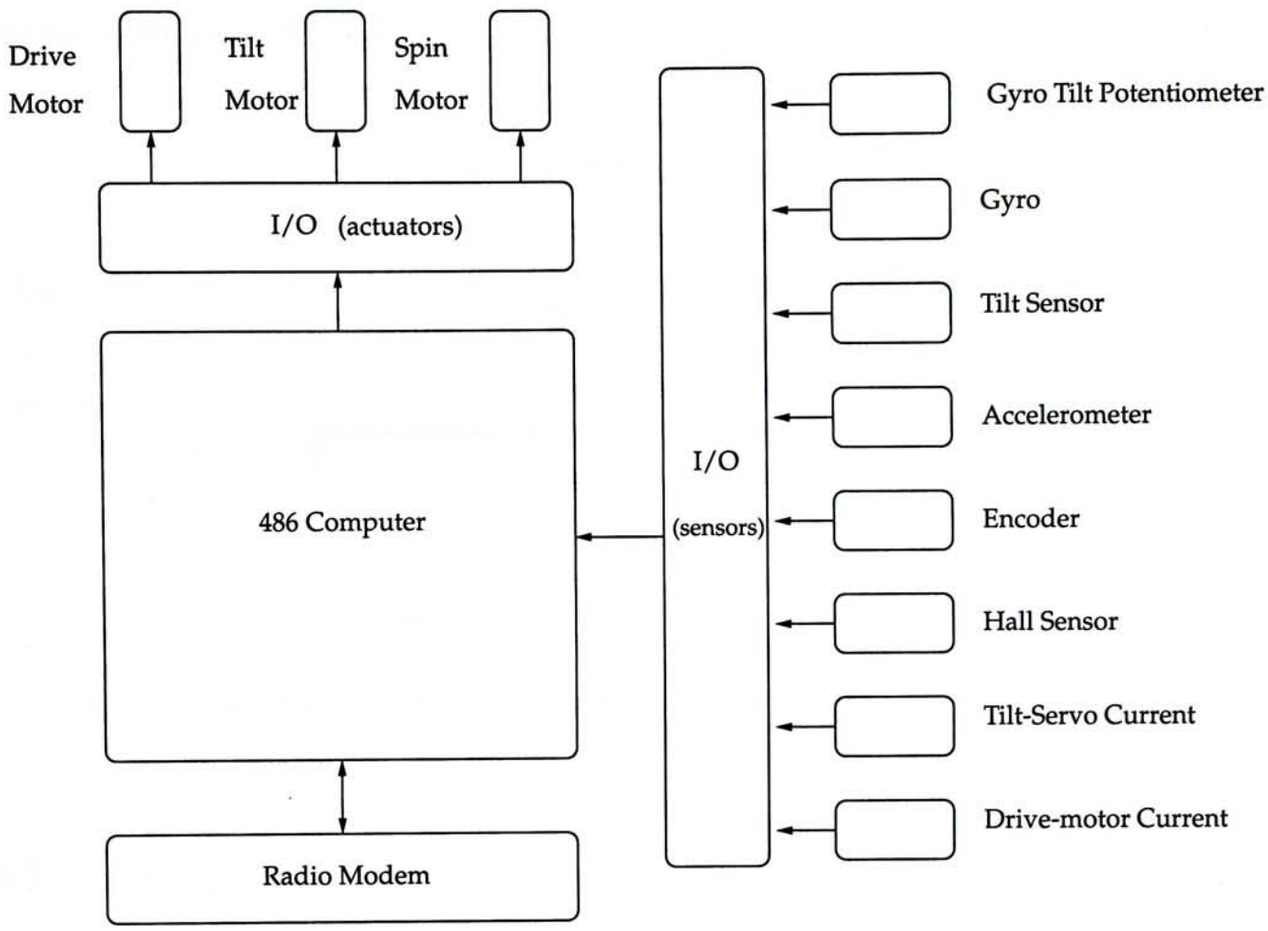

Figure 2.5: **Hardware Architecture of the single wheel robot.**

so that the controller program can access on hardwares quickly and coordinates well on system resources. In addition, client-server programming architecture is applied so that any applications can be easily embedded on the system.

2.3.1 Operating System

QNX, a real-time microkernel operating system is used in our system. QNX is a microkernel which do not contain device drivers. In fact, a device driver is unnecessary for a program to communicate with hardware device. If a user program is given sufficient privilege, it can directly access memory and I/O ports, attach hardware interrupt service routines, etc. This makes writing hardware interface software very easy. Moreover, QNX has very low context switch times (6 μ on a 486 DX2/66) and low interrupt latency (7 μ on a 486D2/66). See [31] for more details

about the architecture of QNX.

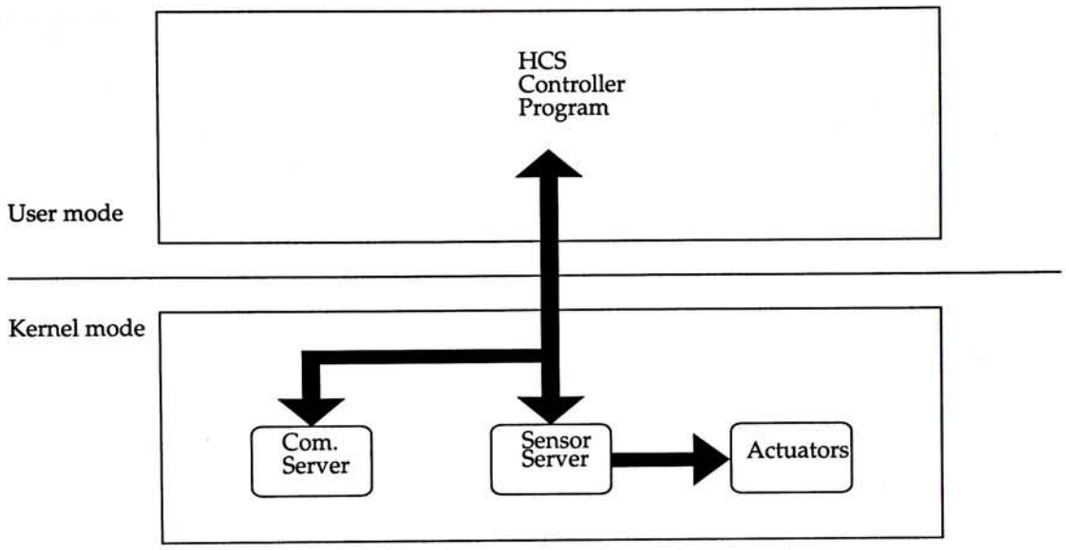

Figure 2.6: **Software Architecture of the System**

2.3.2 Software Architecture

The software system contains three main programs. One program called HCS controller and the other programs are called sensors server and communication server. The sensor server is to acquire the system state by the sensors and control the actuators. The communication server is to provide wireless communication interface. The HCS controller is responsible for computing complicated task and accessing hardware components via sensor server and communication server.

The programs of the robot can be classified into two groups. Any application including HCS controller belongs to user mode. Other programs, which can access hardware components, belong to kernel mode. Sensor server and communication server belong to kernel mode. Figure 2.6 shows the software architecture of the system. Application program must get permission from the kernel when it accesses any hardware component. The advantage of this architecture is that it provides data abstraction and protection of the hardware components. Therefore, any faulty accessing on hardware components by application programs is forbidden. Moreover, any application program can be easily embedded on the system because it

can access hardware resources more conveniently without considering hardware control procedures such as timer interrupt.

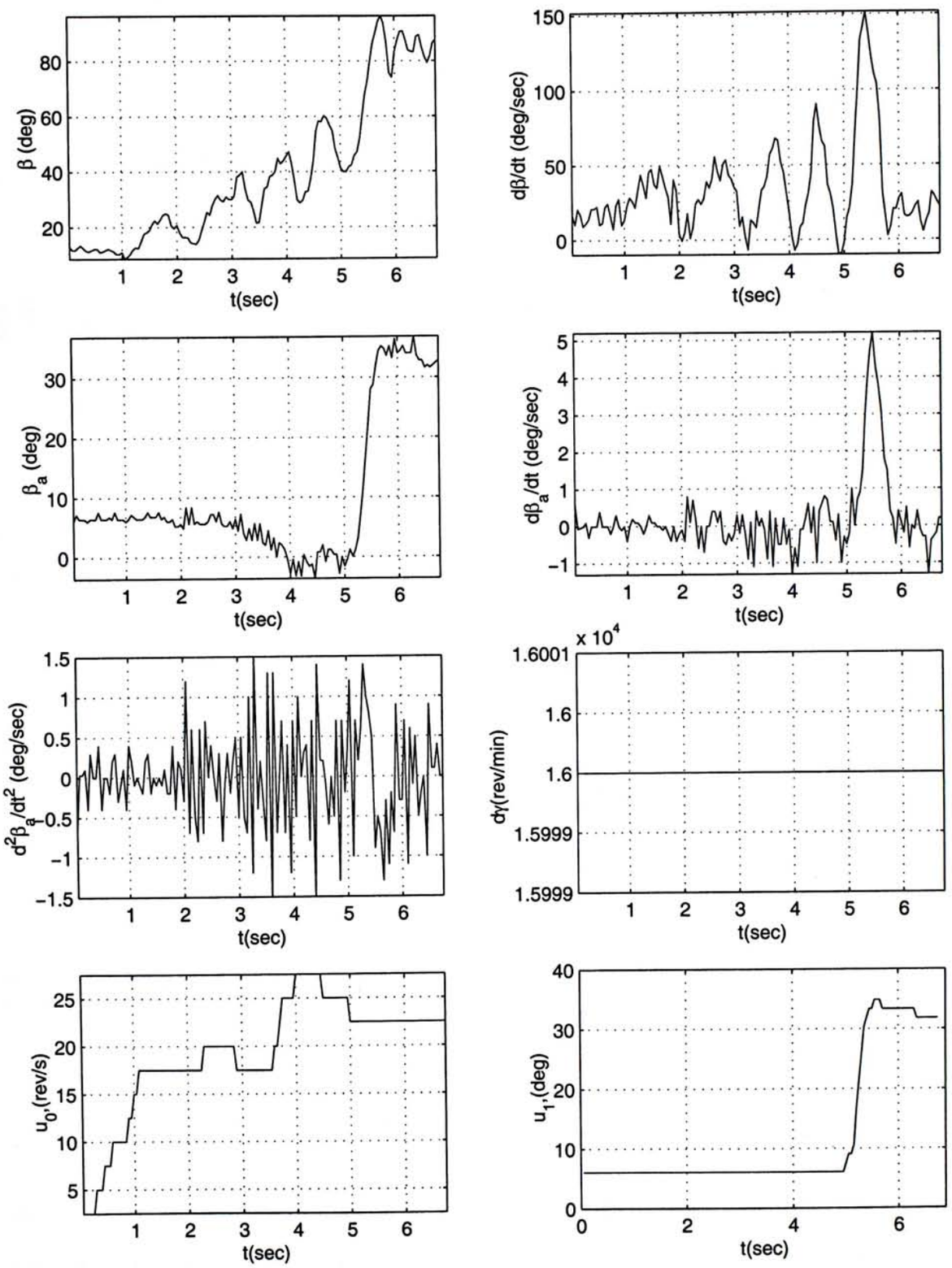

Figure 2.7: **Some sensor data including state variables and control variables of training strategy are illustrated.**

Chapter 3

Human-based Control

In this chapter, we first describe the methodology to abstract human control strategy(HCS). The frame work was developed by Xu and Nechyba [1][6][7]. Then, we model human control strategy (HCS) of the single wheel robot and transfer the HCS model in controlling the robot. We record the operator's control skills and the corresponding states of the robot when he/she drives the robot. We use these data to train HCS model by using the flexible cascade neural network learning architecture with node-decoupled extend Kalman filtering. Then, we transfer the HCS model in controlling the single wheel robot. We experimentally demonstrate that the robot can be automatically controlled by the Human-based controller.

3.1 Why Human-based Control

Owing to its nonholonomic, underactuated and laterally unstable characteristics[2], it is very difficult to design a model-based controller that ensures continuously trajectory tracking. Furthermore, the unmodeled parameters such as friction are relatively more important in the dynamically stabilized system discussed here than a conventional statically stable system, like a 4-wheel vehicle. Although, a mathematical model and model-based controller have been developed for stability action and path following[2]. The model and the controller solely rely on the assumption that the robot rolls without slipping. Thus, some kinds of robot motions cannot be

analyzed by using this model, such as "tilt-up" or sliding motions. In the tilt-up motion, the robot initially lies on the ground. Then, the robot can be "tilt-up" by changing the angular momentum of the flywheel. Because the robot do not roll at the beginning, it violates the above assumption.

On the other hand, humans are capable of mastering complex, control skills on the robot. A human operator can control the robot fluently after operating the robot for a while. Thus, we naturally come up the idea of learning human control strategy (HCS) and develop a human-based controller for the robot. From previous works[1][6][7], modeling HCS can abstract human control skills and transfer in controlling a vehicle. Moreover, human-based controller is a task-based controller that is derived from empirical data of the task, we can develop different human-based controllers for performing different tasks.

3.2 Modeling Human Control Strategy

The following descriptions in this section are the work developed by Xu and Nechyba [1][6][7].

3.2.1 Human Control Strategy

Generally speaking, human skills can be classified into two groups: (1)action skill and (2) reaction skill. Action is the state or process of doing or acting and, the associated skill is called action skill. The main characteristic of action skill is that sensory feedback is unnecessary in the process, or sensory information is unavailable from observation. On the other hand, the characteristics of reaction skill is much more complex. Sensor, decision-making and feedback are necessary for executing reaction skill successfully. Kicking a ball is an example of action kill. Driving a car is an example of reaction skill, where the human closes the feedback control loop. Human control strategy we study in this thesis is a subset of this type of reaction skill. Human control strategy lies between low-level feedback control and high-level reasoning, and has numerous potential applications with a reasonably

well-defined numeric input-output representation.

Human intelligence is complex and difficult to understand at all. We cannot analyze human intelligent analytically via existence of scientific knowledge. Therefore, Modeling human control strategy must rely on observation or learning from empirical data rather than analytical derivation. Because an individual HCS is unique, complex, and unknown properties in nature, we require a learning paradigm which can cope with many difficult challenges.

In the modeling, some factors such as structure, order, granularity and control delay inherent in a particular individual's internal controller are necessary to be considered. However, since each person has his/her unique characteristic, these controller properties can vary substantially from one individual to the others. Structure refers to the functional form which can best approximate the underlying control strategy; order refers the extent to which an individual's control strategy depends on prior histories of sensor inputs and control action outputs; and granularity and control delay quantify the minimum controller time resolution and reaction time for a given individual, respectively.

Moreover, human control strategy is characterized by dynamic and nonlinear natures. Humans are not machines and their actions are prone to errors and gradual changes over time. In addition, human control actions can vary smoothly as well as discontinuously with sensory inputs. Thus, human control strategy is a nonlinear and discontinuous mapping from present and prior sensory inputs and control actions, to future control action outputs.

3.2.2 Neural Network for Modeling

Here we introduce a continuous learning architecture for modeling human control strategy. This architecture is called the cascade neural network architecture (CNN)[16][17]. Cascade neural networks can learn complex, nonlinear HCS mapping from input-output data. Unlike most conventional neural network, the structure of cascade neural network is not fixed before learning begin. The structure of

cascade neural network evolves in learning process. Moreover, the cascade learning algorithm consists of both aspects of function approximation–the selection of an appropriate functional form and the adjustment of free parameters in the functional model to optimize some error criterion. These features are significant for modeling HCS because few prior knowledge is known on the human controller structure.

The cascade neural network can adjust the size of the neural network as part of learning by the following two features.

- Feedforward cascade architecture: Hidden units of the network are automatically added one at a time to an initially minimal network.
- Learning algorithm: It creates and installs new units to reduce the root mean square error e_{rms} between model output and the source human control output.

3.2.3 Learning Procedure

Modeling proceeds in several steps.

1. At the beginning, the network is only direct input-output connections. There is no hidden unit in the network.
2. These weights are trained to reduce the e_{rms}. These nodes and weights describe any linear relationship between the inputs and outputs.
3. When the e_{rms} decreases slowly enough, the first hidden unit is added to the network from a pool of candidate units. Using the quickprop algorithm , these candidate units are trained independently and in parallel with different random initial weights.
4. The best hidden unit is selected from the pool and installed in the network, after no more appreciable error can be further reduced.
5. The weights of hidden unit input are frozen while the weights to the output are retrained.

6. When the e_{rms} is sufficiently small for the training sets, or the number of the hidden units reaches as specified maximum number, the modeling procedure stops, otherwise repeat step 2.

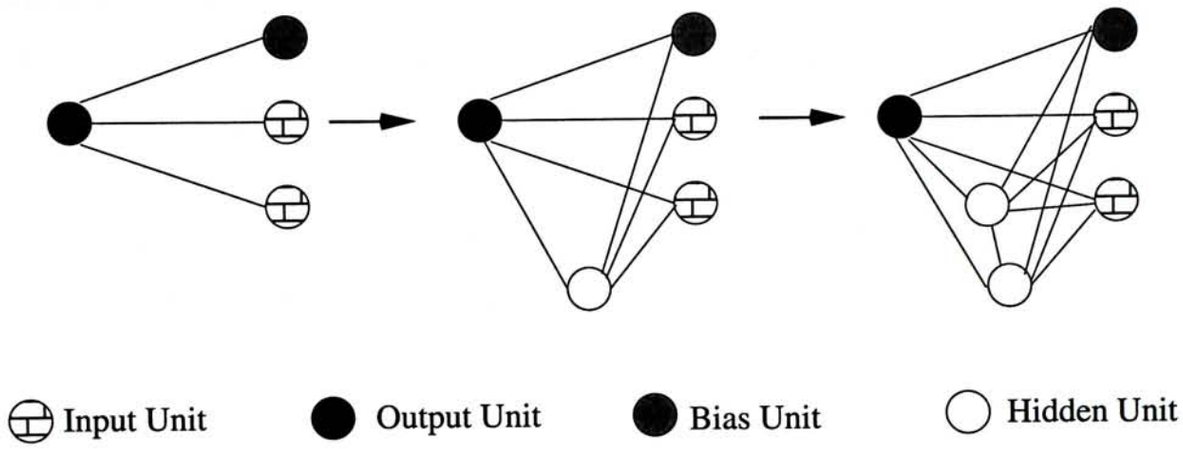

Figure 3.1: **Learning procedure of cascade neural network**

The process is repeated until the algorithm succeeds in reducing the e_{rms} sufficiently for the training set or the number of hidden units reaches a specified maximum number. Figure 3.1 illustrates, for example, how a two-input, single-output network grows as two hidden units are added. Note that a new hidden unit receives as input connections from input units as well as all previous hidden(hence the name "cascade"). A cascade network with n_i input units (including the bias unit), n_h hidden units, and no output units, will have n_w connections where,

$$n_w = n_i n_o + n_h(n_i + n_o) + (n_h - 1)n_h/2 \tag{3.1}$$

Recent theorems done by K. Funahasi[22] and K. Hornik, et. al.[23] proven that standard layered neural networks are universal function approximators. These result can be extended to the cascade network topology. Because a cascade neural network with k hidden units with some weight connections equal to zero is a special case of a multi-layer feedforward neural network with k hidden units arranged in m layers, fully connected between consecutive layer. That is why cascade neural network is used to obtain the model of human control strategy.

The prior assumptions of the functional form of the model, that is structure of neural network, are relaxed by the cascade architecture. These assumptions can be

further relaxed by allowing new hidden units which have variable activation functions. Different variable activations are assigned to the candidate units. During the candidate training, the best candidate which can reduce the most e_{rms} of the training data are selected and installed in the network. Therefore, the candidate unit with the most appropriate activation function at that point during training is selected. Typical alternatives to the sigmoidal activation function are the Gaussian function, Bessel functions, and sinusoidal functions of various frequency.

For training the multi-layer neural networks, the back propagation algorithm is usually employed. The main weakness of this algorithm is that the convergence speed is slow and thus, the corresponding learning time is long. The training time may last for hours or even days when the system is complex, high-dimensional and the resultant network has large number of nodes. The quickprop algorithm is employed to improve the speed of training time over standard back propagation algorithm. It is still a gradient descent based algorithm, which although simple, can require many iterations until satisfactory convergence is reached [15]. Here, the standard cascade learning is modified by replacing the quickprop algorithm with node-decoupled extended Kalman filtering (NDEFK), which has better convergence properties and faster training than gradient-descent techniques for multi-layer feed-forward networks[6][18].

Computational and storage complexity can be reduced by NDEFK as it decouples weights by nodes so that only the interdependence of weights feeding into the same units are considered. In this formulation, the weights in the neural network are considered to represent the state of a nonlinear, finite-dimensional, discrete-time system. The resulting weight update recursion is given by

$$\omega_{k+1}^{i} = \omega_{k}^{i} + \{(\psi_{k}^{i})^{T}(A_{k}\xi_{k})\}\phi_{k}^{i} \tag{3.2}$$

where ω_k^i is the input-side weight vector of length m_i at iteration k, for unit $i \in \{0, 1, \ldots, n_0\}$, where $i = 0$ corresponds to the current hidden unit being trained, and $i \in \{1, \ldots, n_0\}$ corresponds to the ith output unit. ξ_k is the n_0-dimensional error vector for the current training pattern, ψ_k^i is the n_0-dimensional vector of par-

tial derivatives of the network's output unit signals with respect to the ith unit's net input, and

$$\phi_k^i = P_k^i \xi_k^i \tag{3.3}$$

$$A_k = \left(I + \sum_{i=0}^{n_0} \{(\xi_k^i)^T \phi_k^i\} \left(\psi_k^i (\psi_k^i)^T \right) \right)^{-1} \tag{3.4}$$

$$P_{k+1}^i = P_k^i - \{(\psi_k^i)^T (A_k \psi_k^i)\} \{\phi_k^i (\phi_k^i)^T\} + \eta I \tag{3.5}$$

where ξ_k^i is the n_0-dimensional input vector for the ith unit, P_k^i is the $m_i \times m_i$ approximate conditional error covariance matrix for the ith unit, and η is a small number (0.0001) which alleviates singularity problems for P_k^i[18].

Network Model

The class of models we used in this thesis is restricted to static mapping between inputs and outputs. Since human control strategy is dynamic in nature, HCS model depends not only the current sensory information but also previous sensory data. For example, figure 3.2 illustrates how this is done for a SISO system of the form. In general, any unknown dynamic system can be approximated by providing a time history data[28].

We can approximate a dynamic system by a difference equation of the general form,

$$u((k+1)\tau) = \Gamma \begin{pmatrix} u(k\tau), u((k-1)\tau), \ldots, u((k-n)\tau), \\ x((k+1)\tau), x(k\tau), \ldots, x((k-m)\tau) \end{pmatrix} \tag{3.6}$$

where $\Gamma(\cdot)$ is some arbitrary unknown function, $u(k\tau)$ is the control vector, $x(k\tau)$ is the system vector at time instant k and τ indicates the controller resolution or granularity. The order of the dynamic system is given by the constants n and m, which may be infinite. In this thesis, we assign $n = 4$ and $m = 3$.

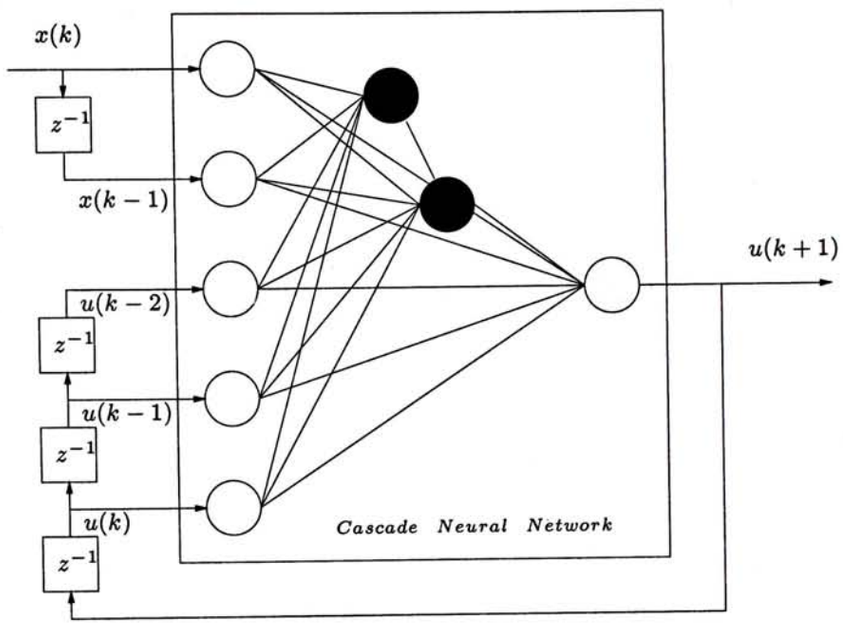

Figure 3.2: **A dynamic system can be mapped into a static cascade neural network by providing a time history of data as input to the network**

3.3 Task Descriptions

In our experiment, we study the "tilt-up" motion of the single wheel robot and hope to develop a human-based controller for automatic "tilt-up" motion. Our goal of "tilt-up" is to have : lean angle $\beta \approx 90^o$. We want to demonstrate that HCS model can abstract human tilt-up skills and corresponding human-based controller is capable of controlling the robot with similar skills.

In our system, there are three control variables, u_0 controlling rolling speed of single wheel $\dot{\gamma}$, u_1 controlling the angular position of flywheel β_a and u_2 controlling spinning rate of flywheel $\dot{\gamma_a}$. We do not model u_2 simply because the control variable u_2 is usually kept at constant.

At the beginning of the task, the states of the robot is shown in figure 3.3 (a). Then, the robot starts to roll as u_0 increases. By changing the angular momentum of the flywheel as shown in figure 3.3 (b), a tilting torque is acted on the system. As a result, the robot can tilt up as shown in figure 3.3 (c). The human skill in controlling the tilt-up motion strategy is shown in figure 3.4.

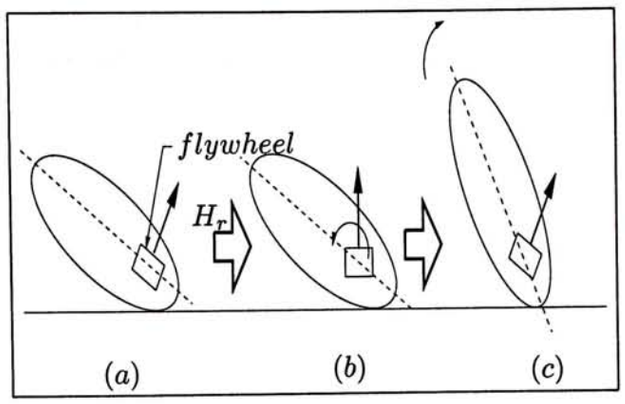

Figure 3.3: **The interaction of the Wheel and flywheel during tilt-up motion.**

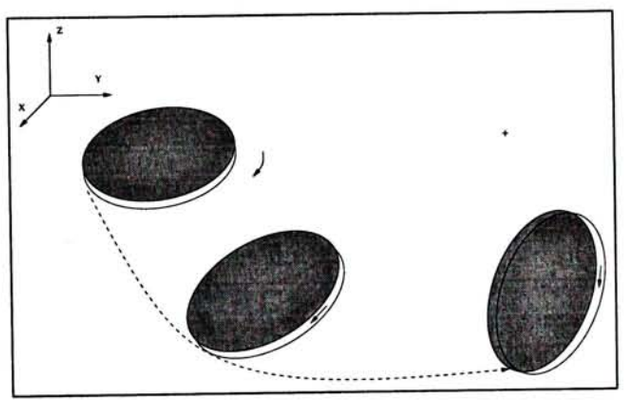

Figure 3.4: **Tilt-up motion strategy of Gyrover.**

3.4 Modeling HCS in Controlling the Robot

Modeling HCS means to model the human operator's control action in response to the system real-time feedback. That is why we use the cascade neural network to construct the mapping between the dynamic human control action (control commands) and the system response (state variables). A dynamic HCS can be approximated by the difference equation[28]. Thus, the human control strategy U_h at the time instant $k+1$ can be approximated by equation 3.6. The equation 3.6 considers the previous state variables and control commands because these previous variables are significant for human operators to control the robot. The cascade neural network with extend node-decouple Kalman filtering is employed to abstract the input-output relationship in the equation 3.6.

Figure 3.5: **Human operator tilts up the single wheel robot at the beginning .**

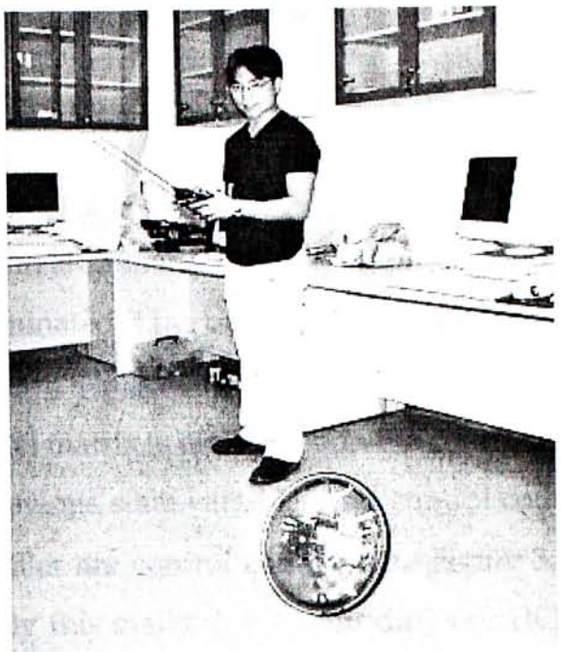

Figure 3.6: **Human operator tilts up the single wheel robot at the end.**

3.4.1 Model Input and Output

The model input is defined as the combination of present and prior state variables and control commands (see section 3.4). The model output is the present control commands. Here, we want to model human control skill on drive motor u_0 and tilt motor u_1. Then the control commands for model input are u_0, u_1 and we have to select a set of data out of all available sensing variables. For convenience, the set of state variables for model input is called input vector. And the selected state variable is called input variable. In selecting potentially relevant state variables, we first would like to select a set of variables which are important for an operator to make a decision. Second, they must be measurable using the on-board sensors, and at the same time, it must be visually observable for the operator. In fact, there are over 12 sensor values available combining their derivative. Therefore, some of data must be redundant because a operator is impossible to make decision based on numerous state variables. Preliminarily, we intuitively select the state variables $\{\beta, \beta_a, \dot{\alpha}, \dot{\beta}, \dot{\gamma}, \dot{\beta}_a\}$ as input vector, based on the mathematical model in [10] and the driving experience for the robot. Then, we use these input-output data to train HCS

models.

3.4.2 Human-based Controller

We use the learning rule (equation 3.5) to train HCS model with the empirical input-output data. The learning procedure is terminated if the output e_{rms} is sufficiently small. The resultant model is a $(n_O + n_h) \times (n_i + n_h)$ matrix. To develop a human-based controller from HCS model, the model matrix is embedded in the controller. The controller requires the present and previous state variables and control commands as input. The output of the controller are control commands. Figure 3.7 shows the control diagram of the robot. By this method, we train different HCS models and transfer to human-based controllers. Then, we use these controllers to control the robot. In next section, we will experimentally demonstrate the performance of these controllers.

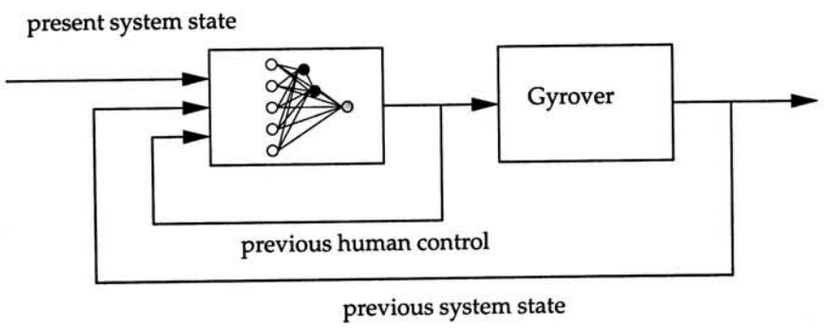

Figure 3.7: **System diagram of human control strategy model.**

3.5 Result and Discussion

Figures 3.9 - 3.15 show the experimental results for the automatic "tilt-up" motion by using human-based controllers. In the figures, four important variables (β, β_a, u_0, u_1) are listed in the examples where β is the lean angle representing target of the mission, β_a is the tilt angle of flywheel generating main tilt torque force for stabilization, and u_0 and u_1 are the drive torque and tilt torque respectively

In each experiment, the goal can always achieve: $\beta \approx 90^o$ even if the initial an-

gles and the execution time varies. It demonstrates that the tilt-up skill we selected for learning is valid. The learned control input is capable of controlling the dynamically stabilized robot.

In our experiment, it is difficult to keep the initial lean angle β to be exactly same as that when human manually control it. However, the result seems to be insensitive to the slightly different initial lean angles we set. The perturbation of the lean angle under the learned control input is relatively small. Similar case can be found in the figures 3.10-3.11, the lean angle β are overshooted over 10 degrees. It is interesting to note that u_0 becomes small while u_1 become large in the figure 3.11 . When the tilt torque increases significantly, u_0 could reduce the tilt torque, based on equation 1.1. It shows the learned HCS model have a high adaptability for stabilizing the system by reducing overshooting.

From the results of the learned model, the system is unstable initially, especially on the figures 3.13,3.14,3.15. There are two major reasons for the unstable motions. Firstly, the initial position is largely different from that in the training phase. The range of initial lean angle β of the learned model is within $[10^o, 30^o]$ while the initial lean angle β of the source human data is $\approx 18^o$. It is difficult for the HCS controller to classify the situations and make a suitable reaction in the initial phase. Secondly, sliding occurs as the tyre of the robot does not touch on the ground initially. It is one of weakness of the HCS controller as no sensor is available to detect the friction on testing platform. Since our testing environment is located at outdoors, it is very difficult to guarantee that the friction remains constant in the whole testing region. Considering the figures 3.13,3.14,3.15, the control vector u_0, u_1 fluctuate in the initial state and also make the wheel and flywheel vibrate. Nevertheless, the single wheel robot is unstable at that moment, the learned model could control the motion properly and the goal is achieved finally. Thus it shows that HCS controller has learned the human tilt-up skill and transferred in controlling the robot.

In the experiments, the total time of the learned models for completing the task are slower than that of the source human data. The average total time for the learned models are approximately around 15s while the total time of the training data is around 6.5s. It is because the HCS controllers spend lot of time for com-

puting. Sensor data are acquired at 50ms sampling rate when the human operator controlled the robot. When the learned model is used, the running clock cycle is set to 100ms sampling rate in order to have enough time for computing the HCS controller. Thus, the response time of the learned model seems to be slower. Moreover, acquired data contain different degree of noise. The HCS controller could not classify the situation easily and make a suitable response. Similarly, when human receive unexpected data, he/she will hesitates and respond slowly like the robot.

In this chapter, we presented a method of modeling human operator 's strategy in controlling a dynamically stabilized robot. We first selected relevant state variables from dynamics equations. We experimentally implemented the method and demonstrated that the robot could be automatically controlled using the learned human control input. The work is significant to abstract human control strategy for controlling a dynamically system in generating a automatic control input.

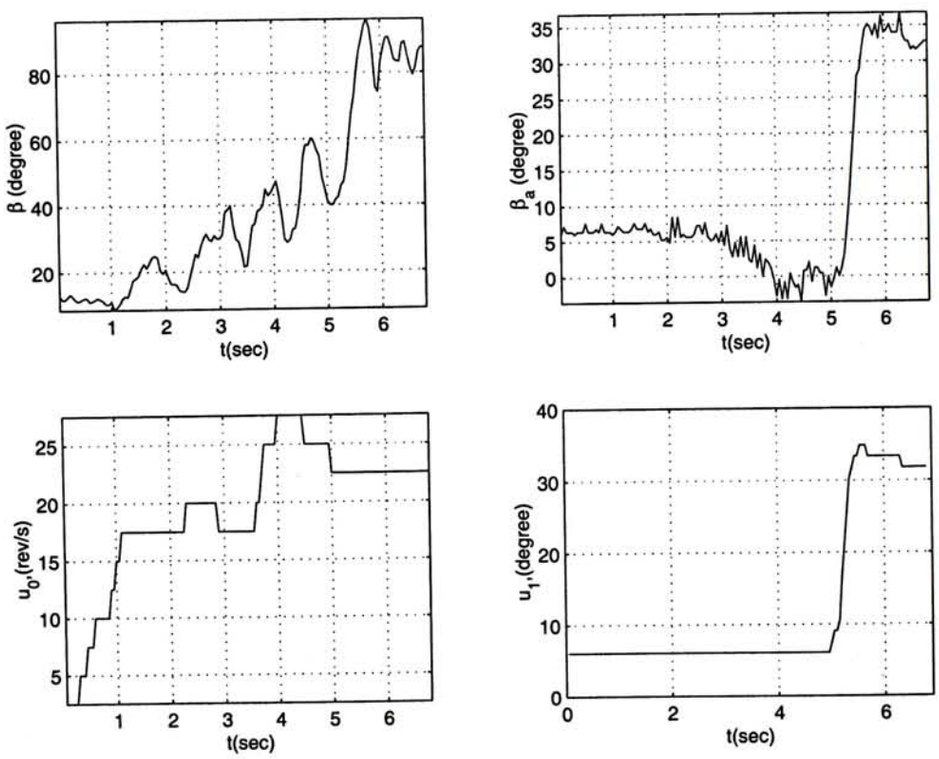

Figure 3.8: **Operator's skill in controlling the tilt-up motion**

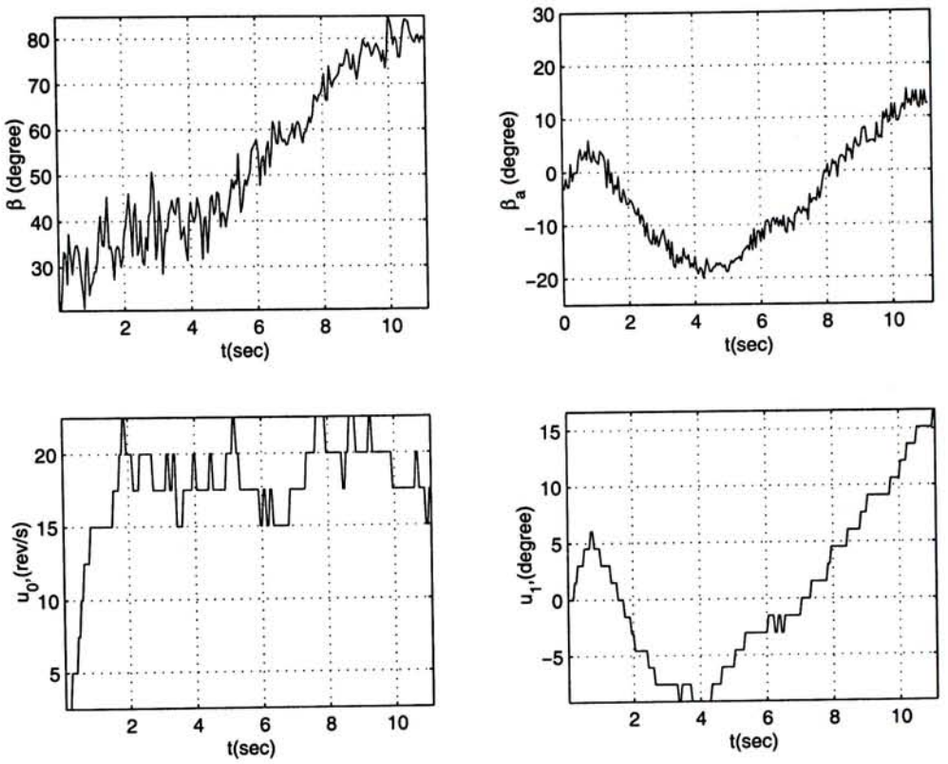

Figure 3.9: **Experimental result done by human-based controller: case I**

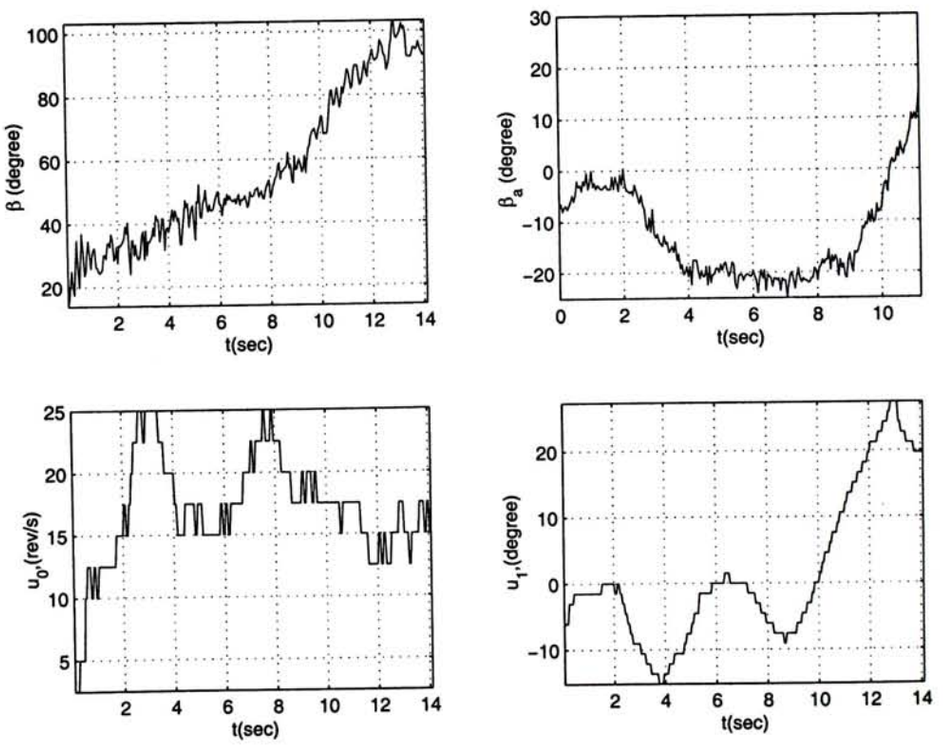

Figure 3.10: **Experimental result done by human-based controller: case II**

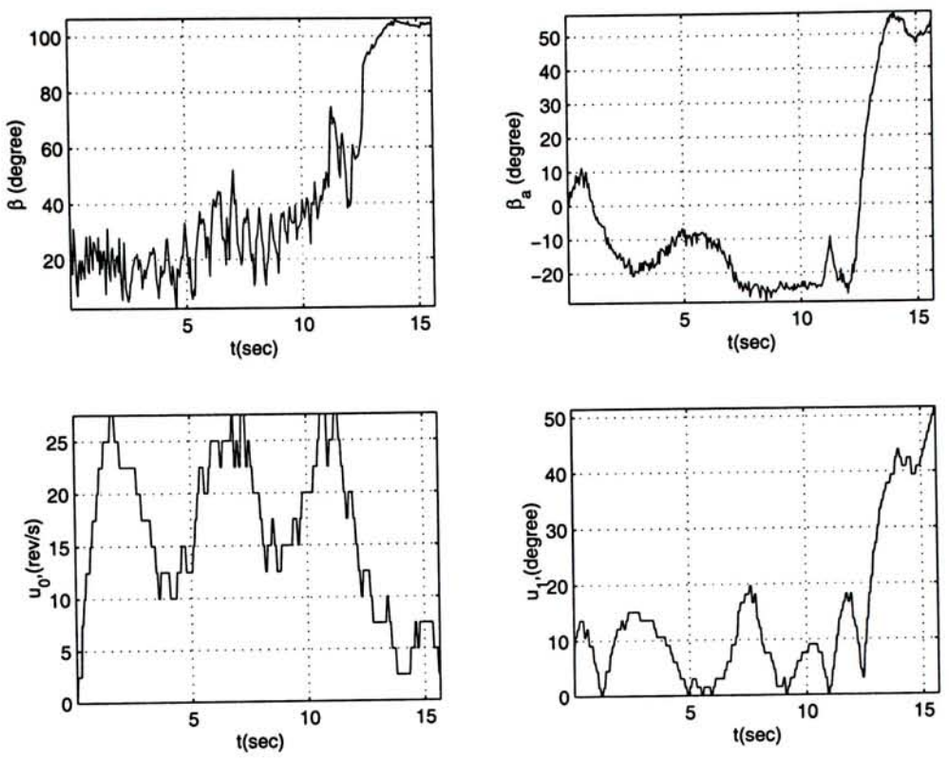

Figure 3.11: **Experimental result done by human-based controller: case III**

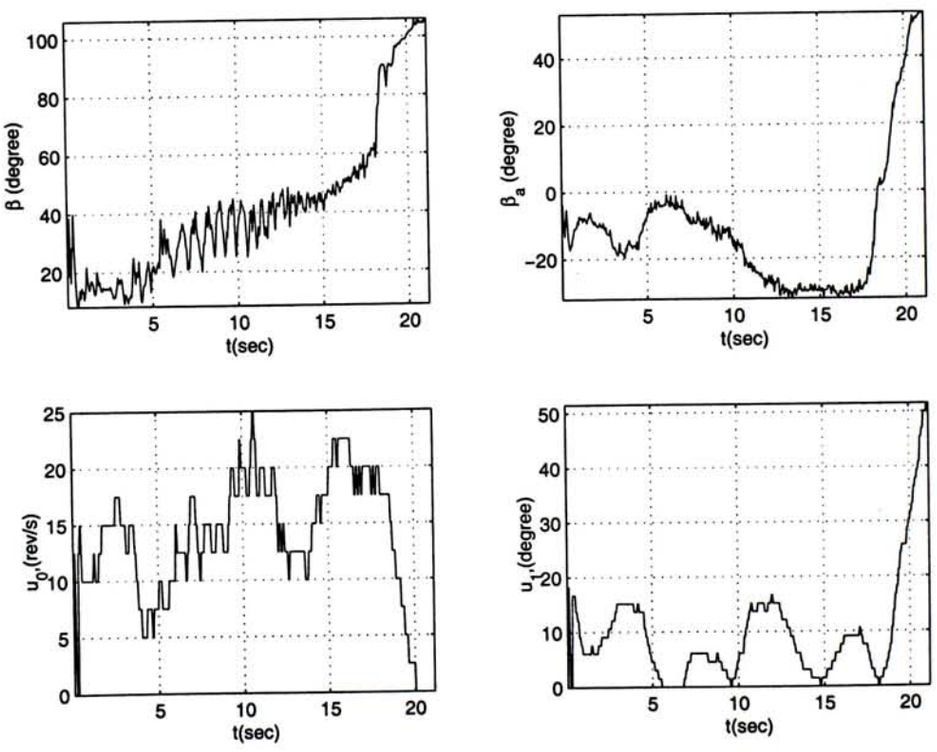

Figure 3.12: **Experimental result done by human-based controller: case IV**

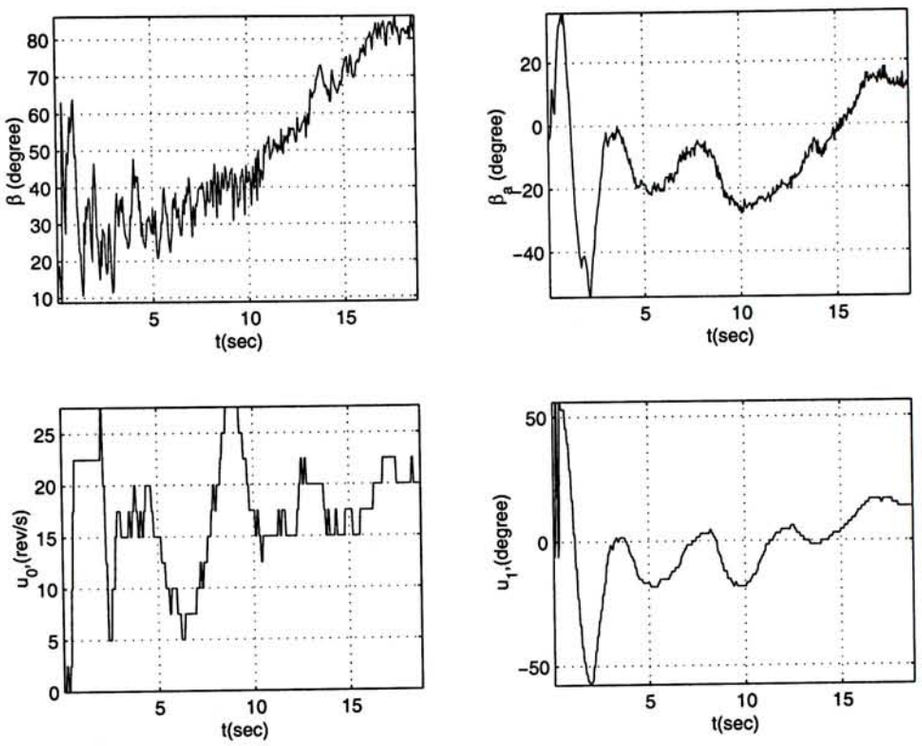

Figure 3.13: **Experimental result done by human-based controller: case V**

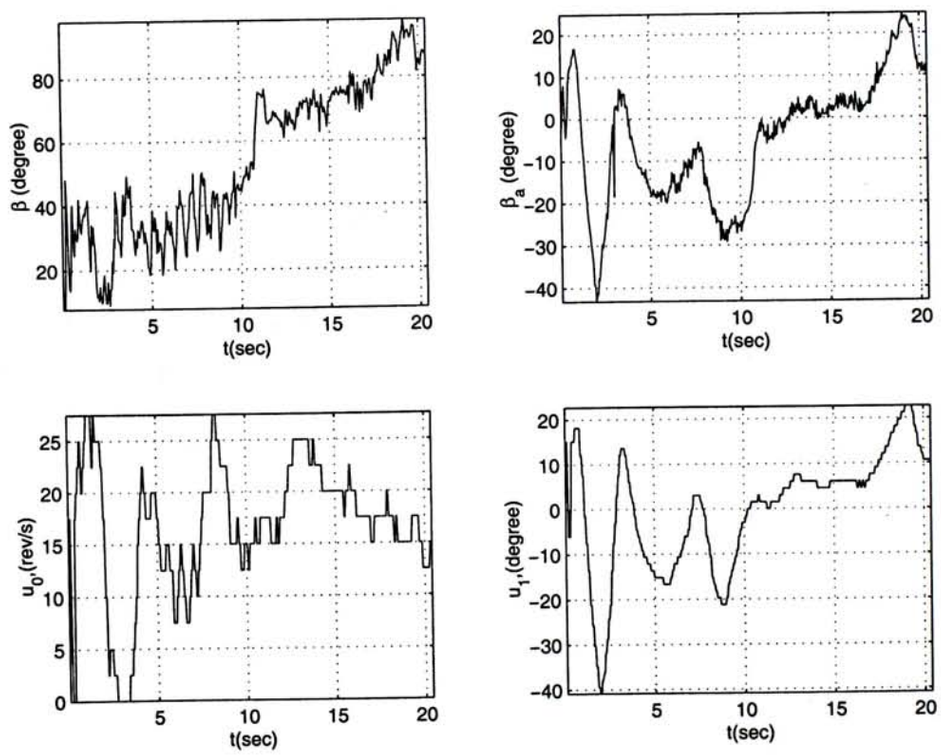

Figure 3.14: **Experimental result done by human-based controller: case VI**

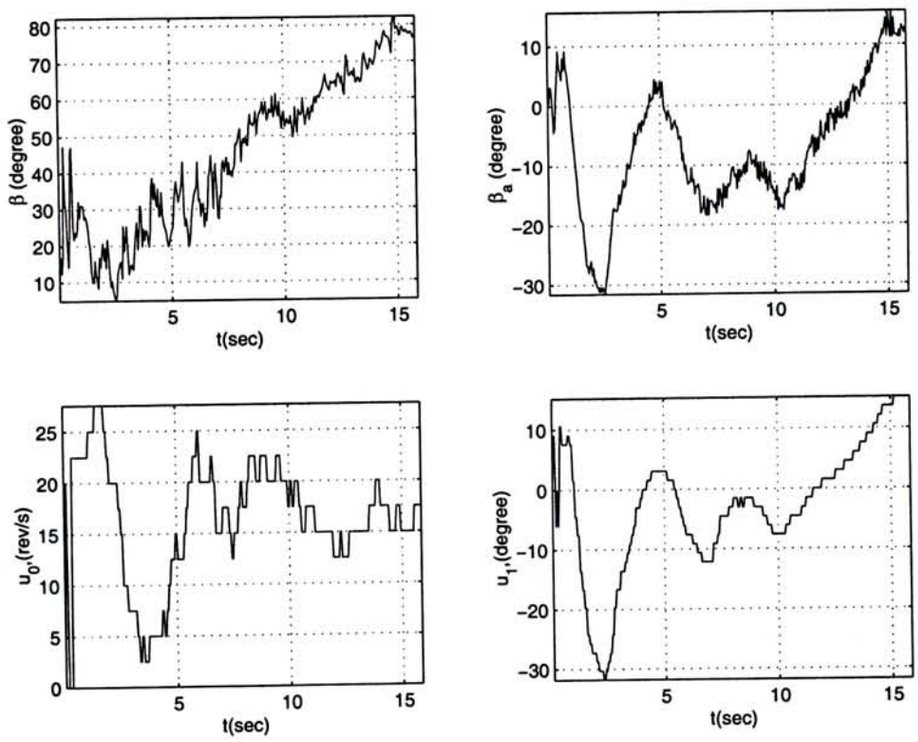

Figure 3.15: **Experimental result done by human-based controller: case VII**

Chapter 4

Input Selection

In the system, there are numerous state variables available for input selection. Moreover, sensor data contain different degree of irrelevant, noisy information and thus, influence the learning process. Thus, the selection of model input plays an important role for extracting human skills.

Broadly speaking, input selection can be categorized into two groups: *model-based* approach and *model-free* approach [46][47]. Model-based approach typically involves selecting a model, choosing the inputs to use, and then measuring the performance. Model-free approach is based on performing a statistical test between the subsets of input variables and the desired output from the model. Here, we develop two selection methods to select suitable input variables to handle the selection problem. In the first method, we select the input variables from the mathematical model of the robot. In the second method, we select the input variables based on a statistical tool, *factor analysis*.

4.1 Why Input Selection

Human control data is acquired by sensors to train HCS model. There are numerous sensor variables and their corresponding derivatives available from the systems, e.g. Euler angles (roll-pitch-yaw) of the robot and corresponding angular velocities and angular accelerations. Moreover, irrelevant data will affect the mod-

eling HCS while the learning performance can be accelerated by eliminating redundancies in large, unprocessed human data [32]. Also, some sensor data contain high noise and thus, affect the learning process. Furthermore, as the input dimensionality increases, the computational complexity and memory requirements of the model increase.

In addition, to control a highly dynamic system such the single wheel robot, selecting effective input state variables for the modeling HCS is relatively more important than a classically dynamical system such as car. Because the robot largely depends on various unmodeled parameters, such as friction.

What variables should we select as input variables for the modeling? We do not know the best input representation a priori, as it will vary from one individual to the next. Thus we are interested in identifying the combination of present and prior state variables and control inputs upon which a human operator relies most heavily to form his control strategy.

4.2 Model Validation

4.2.1 Why Model Validation

In this chapter, we concern on effect of different model input on the HCS modeling. Therefore, we need to train different HCS models from different combination of input variables and compare the performance of HCS models. In order to compare the HCS models effectively, we need to define a measure to evaluate the performance of the models

HCS models are derived from empirically data by using cascade neural network. The main advantage of the modeling HCS is that no analytical model is required. HCS models are not restricted by the limitation of current scientific knowledge as we do not have adequate understanding of human intelligence at all. On the hand, no explicit physical model is the main disadvantage of modeling because the lack of scientific justification detracts from the confidence. As a result, we cannot compare different HCS models from the parameters of the models. This

problem becomes serious if the unmodeled process is dynamic in nature. In a dynamic process, the modeling errors feed-back on themselves to generate deviations in state, and command trajectories are thus uncharacteristic of source processes.

4.2.2 Root Mean Square Error Measure

Similarity measure for validating human control strategy models has been developed by Nechyba and Xu. This method is based on Hidden Markov Model (HMM) which has been used in many applications, especially a human control data, it can be used to evaluate stochastic similarity between two dynamic multi-dimensional trajectories using HMMs analysis. The similarity measure can be used to validate the model-generated trajectories and the source HCS. By this method, the best learned-model can be selected which has highest scores of similarity measure for HCS controller.

However, this measure is not suitable for our system. Because similarity measure addresses on comparing long, mult-dimensional, stochastic trajectories. In our cases, the control skills is a short-time, static trajectories. A example of control skill trajectories (u_0, u_1) shows in figure 2.7. In the Nechyba's work[1], the total time of control skills is around 20 minutes while the total time of driving the single wheel robot is around 10 seconds. We found that the results of the measure were inaccurate when we evaluated the HCS models. Thus, we purpose to evaluate the output trajectories withe the source skill by a simple approach, root means error measure. The root means square error e_{rms} measures the difference between the training data and predicted model output. It is adequate to gauge the fidelity of a learned model to the source process. Moreover, e_{rms} can serve as the test of convergence. By this way, we can evaluate the performance of the HCS models.

4.3 Experimental Setup

In our experiments, we have a choice to evaluate the learned models in *real* environment or in *simulation* environment. In the previous chapter, we demonstrated

that the human-based controller could control the robot. We collected those control data which were shown in figure 3.9 and compared these control trajectories with the source control trajectories. From this method, we can practical examine the performance of HCS models. However, from the figures 3.9-3.14, we found that the results of output trajectories, which generated by the same human-based controller, varied greatly (see figures 3.9-3.14). That's mean that the human-based controller has low repeatability in practice and thus, we cannot compare the different HCS models by this approach. On the other hand, we can use the source training data as testing data to simulate the input variables and then, HCS model generates the corresponding trajectories. Next, we compare the output trajectories with the source control trajectories. Unlike the first approach, by this method, the HCS models can generate stable output trajectories and it is easy for comparison.

The performance of the models vary for the same model input because the initial weight units of the neural network are randomly assigned and cause different result of learned models at the end of modeling. In order to evaluate accurately the effect of the input vectors, we trained 30 trials on each model input and measured the average $e_{\bar{rms}}$ errors of the output of the models.

4.4 Model-based Method

At the beginning of this chapter, we have introduced two input selection approaches. In this section, we will select input variables by model-based approach. We will first select input variables from the mathematical model of the robot since these variables can describe the motion of the robot. Then, we define a measure of the sensitivity of each state variables with respect to operator's control input to verify the importance of these variables for the modeling.

4.4.1 Problem Definition

Modeling human control strategy means to model the human operator's control action in response to the system real-time feedback. That is, we need to develop a relationship between the human control actions (control commands) and the system responses (present and previous state variables and control variables) as shown in figure 4.1

The HCS model can be simply interpreted as a mapping function Φ such that :

$$\Phi : \mathcal{M} \longrightarrow U_h \tag{4.1}$$

where $\mathcal{M}$ is the model input and U_h is the output control. The HCS model means that the function Φ is derived by empirically input-output data. The input data are system real-time feedback and the output data are control variables. Three factors are necessary to be considered for the learning process. They are (1) input representation, (2) consistent mapping from input space to output space and (3) learning algorithm. As we have mentioned in the chapter 3, HCS is a type of reaction skill and HCS model requires sensory inputs to execute control commands. However, it is possible that same input pattern corresponds to two different output because of missing information [13]. Therefore, input variables should be effectively enough to describe the motion of the robot. As a result, the model can more easily identify the situation and then it can do a suitable reaction from the input data. Secondly, sensory data contain irrelevant, noisy information and influence the learning process. The first and second factors result in discontinuous mapping as shown in figure 4.2(a). Thirdly, the learning algorithm we used have been well-developed by Nechyba and Xu [1][6][7]. Thus, we will not focus on the learning algorithm in this thesis.

4.4.2 Input Representation

In this section, we will focus on selecting a number of state variables for model input. Model Input is defined as the combination of *present and prior state variables and control commands* (see section 3.4). We select a set of suitable state variables (as

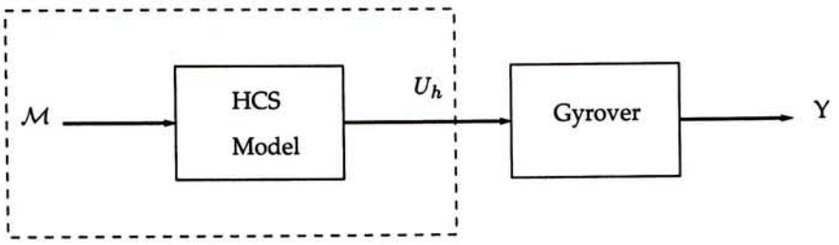

Figure 4.1: **The flow diagram of input-output relationship of HCS model**

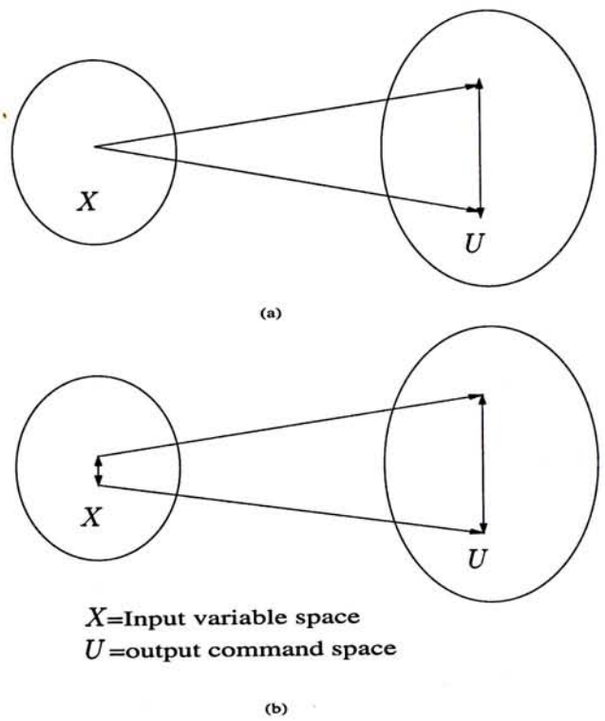

Figure 4.2: **Inconsistent mapping from state variable space X to output control command space U**

input vector) and then form the model input to train HCS model.

The performance of HCS model degrades significantly due to insufficient learning data. Therefore, we should select a set of state variables as the input variables without missing skill information. The basic requirement is that the set of state variables can describe the motion of the robot. What variables are effective enough to describe the motion of the robot? The mapping from input state space to output command space should be injective mapping. It means that any input state vector should have its unique corresponding control action. Otherwise, two or more actions caused by same input state vector (figure 4.2 (a)).

In our approach, we first select state variables intuitively with the aids of kinematic constraints and dynamic equations of the robot derived in [10]. From equation 2.4 in chapter 2, we let $U_h = \begin{bmatrix} u_0^h & u_1^h \end{bmatrix}^T$ be the human control commands. As a result, the input-output of the HCS model must be governed by the equation 2.4 such that

$$\bar{M}(q)\ddot{q} = \bar{F}(q, \dot{q}) + BU_h$$

The equation (2.4) can be re-written in this form:

$$BU_h = \bar{M}(q)\ddot{q} - \bar{F}(q, \dot{q}) \tag{4.2}$$

The above equation shows that the human control strategy U_h can be governed by the state variables $q, \dot{q}, \ddot{q}$. The equation 4.2 can be further re-written into this form:

$$U_h = \Psi(X) \tag{4.3}$$

where X is the input vector such that $X = \begin{bmatrix} q & \dot{q} & \ddot{q} \end{bmatrix}^T = \begin{bmatrix} \alpha & \beta & \gamma & \beta_a & \dot{\alpha} & \dot{\beta} & \dot{\gamma} & \dot{\beta}_a & \ddot{\alpha} & \ddot{\beta} & \ddot{\gamma} & \ddot{\beta}_a \end{bmatrix}^T$ and Ψ is an nonlinear mapping function. The equation 4.3 shows that the input variables $q, \dot{q}, \ddot{q}$ provide enough state information to control the robot. It is easy to show that the Ψ function is a injective mapping. i.e. if $\Psi(X_1) \neq \Psi(X_2)$ then $X_1 \neq X_2$. Therefore, the input vector X is effectively enough to describe the motion of the robot. For the discrete case, the equation (4.3) is in this form :

$$U_h(k) = \Psi(X(k)) \tag{4.4}$$

where k is time instant.

By above result, all variables $x_i \in X$ should be selected as the input vector to form model input. However, we cannot measure α and γ because of unavailability of sensors. In the experiments, our goal is to tilt up the robot until the robot is vertical. Therefore, we only concern the lean angle and thus, the angular position α, γ are relatively unimportant as we do not care the angular position of α, γ in the experiments . Thus, the resultant input vector $X = \begin{bmatrix} \beta & \beta_a & \dot{\alpha} & \dot{\beta} & \dot{\gamma} & \dot{\beta}_a & \ddot{\alpha} & \ddot{\beta} & \ddot{\gamma} & \ddot{\beta}_a \end{bmatrix}^T$ is selected for the modeling.

4.4.3 Sensitivity Analysis

From above result, we use the input vector to form the model input. Model input consists of the prior and present input vector and control vector. Here, we set $n = 4,\ m = 3$ in the equation 3.6 and thus the model input is $[X(k+1), X(k), X(k-1), X(k-2), X(k-3), U_h(k), U_h(k-1), U_h(k-2), U_h(k-3), U_h(k-4)]$(equation 3.6) and thus, the total dimension of model input is 60. The input dimension is large and consequently, the corresponding computational complexity and memory requirements of the model increase. Therefore, we want to reduce the input dimension by eliminating less important variables.

From the mathematical model, we have found a number of state variables as the input vector and the mapping between $X(k) \rightarrow U_h(k)$ is injective. However, in practice, there exists inconsistent mapping [13] as shown in figure 4.2 (b). It means that human makes different control actions U on similar model inputs. In other words, very similar inputs can lead to radically different outputs U. Thus, the control strategy is approximately discontinuous as human have different responds for similar input pattern. Consequently, the control strategy may not be easily expressible in a discontinuous function. This poses an impossible learning challenge not just for cascade neural networks, but any continuous function approximator. In theory, no continuous function approximator will be capable of modeling this discontinuous strategy.

The cascade network is difficult to extract the discontinuous mapping as the resulted models do not appear to exhibit a high degree of fidelity to the source human data[1]. We make a hypothesis that if the individual variable has discontinuous mapping to the output, it is difficult for the network to extract the mapping. Therefore, we can get rid of this variable from the input vector because the effect of this variable for learning is weak. We will verify this hypothesis in the next section. In order to reduce the dimension of model input, we define a sensitivity function as follow:

$$T(x_i(k)) := \frac{||\Delta U_h(k)||_2}{||\Delta x_i(k)||_2} \tag{4.5}$$

where $X = [x_1 \ x_2 \ \cdots]^T$ is a set of the state variables we selected, and $U_h = [u_1 \ u_2 \ \cdots]^T$ is a set of the control commands. Both X and U_h are time functions such that $X(k) = [x_1(k) \ x_2(k) \ \cdots]^T$ and $U_h(k) = [u_1(k) \ u_2(k) \ \cdots]^T$ representing $X(k)$ and $U_h(k)$ at time $t = k$.

The idea of the sensitivity function comes from the gradient of control command U_h.

$$\nabla U_h = [\frac{\partial U_h}{\partial x_1} \quad \frac{\partial U_h}{\partial x_2} \quad \cdots] \tag{4.6}$$

If $\frac{\partial U_h}{\partial x_i} = \infty$, the output U_h is highly sensitive to input variable x_i. By this way, we extent the idea, thus, if $||\Delta x_i(k)||_2 = ||x_i(k) - x_i(k-1)||_2$ decreases, while $||\Delta U_h(k)||_2$ just changes much relatively, i.e., $T(x_i)$ becomes extreme large, it implies that the output command is very sensitive to the input state variables x_i and the network is difficult to extract this input-output relationship. Therefore, we give up this redundant variable x_i as the state variables for model reduction. The sensitivity analysis of input variables must be bounded:

$$\frac{||\Delta U_h(k)||_2}{||\Delta x_i(k)||_2} < L_u \tag{4.7}$$

where L_u is upper bound value. We can only select those state variables fulfilling the criterion.

In the experiments , preprocessing process, which proceeds the sensitivity analysis, contains two steps: normalization and moving average [1]. Firstly, we normalize X and U_h. The moving average of X is computed before the sensitivity analysis. Moving average is to reduce the perturbation of the acquired noisy data.

Table 4.1 illustrates the average value of sensitivity measure $T(x^i)$ on each sample. The $T(x_i)$ which has highly sensitive value (> 100) are underlined in the table. Here, we define the sensitive value which is greater than 100 is called highly sensitive value. For example, $(\beta_a, \dot{\beta}_a)$ are not selected as the model input in the sample 1. $(\dot{\beta}_a, \ddot{\beta}_a)$ are not selected as the model input in the sample 2. In addition, we cannot find any extreme high value in sample data 5,6 from the sensitivity analysis.

[1] A moving average of order N is simply the arithmetic average of the most recent N observation, $\bar{x}(k) = \frac{x(k)+x(k-1)+\cdots+x(k-N)}{N}$.

Table 4.1: **Sensitivity Analysis on human tilt-up skill data**

average $T(x_i)$	$T(\bar{\beta})$	$T(\bar{\beta}_a)$	$T(\bar{\dot{\alpha}})$	$T(\bar{\dot{\beta}})$	$T(\bar{\dot{\gamma}})$	$T(\bar{\dot{\beta}}_a)$	$T(\bar{\ddot{\alpha}})$	$T(\bar{\ddot{\beta}})$	$T(\bar{\ddot{\gamma}})$	$T(\bar{\ddot{\beta}}_a)$
sample 1	4.416	13×10^3	3.022	2.561	1.690	14×10^3	1.566	1.099	13.89	2.751
sample 2	7.688	11.89	16.68	0.729	1.165	38×10^3	2.366	4.228	2.082	19×10^3
sample 3	2.714	850.5	6.132	4.725	4.270	3.428	4.259	4.516	1.915	5186.6
sample 4	4.159	10.06	1.026	6.510	1.540	3×10^3	1.083	7.944	0.938	613.4
sample 5	1.884	5.314	3.166	1.805	3.602	3.228	2.161	1.331	1.341	0.980
sample 6	19.91	23.50	2.906	1.381	15.21	1.646	3.421	4.711	3.757	2.601
sample 7	19.43	8.464	6.395	1.244	4.937	5×10^3	3.549	3×10^3	2.883	685.7
sample 8	24.14	11.83	3.280	7×10^3	9.574	2.547	8.164	6×10^3	7.718	1×10^3

By this method, we can eliminate undesirable variables for modeling HCS. In next section, we will validate the claim of sensitivity analysis by comparing different model inputs and their corresponding output.

4.4.4 Experimental Result

In this subsection, we demonstrate the performance of the HCS models trained with different combination of input variables. First, the input vector from 8 sets of source human tilt-up skill data are used to train HCS model. Then, we compare the models trained by input vector $\{\beta, \beta_a, \dot{\alpha}, \dot{\beta}, \dot{\gamma}, \dot{\beta}_a, \ddot{\alpha}, \ddot{\beta}, \ddot{\gamma}, \ddot{\beta}_a\}$ (selected from section 4.4.2) with this vector without highly sensitive variables in each human skill data. For further comparison, we select input vector without lowest sensitive variable in the table 4.1 to demonstrate the influence of different sensitive degree of state variables on the modeling. For convenience, X represents the input variables selected from section 4.4.2 , X_l represents the input variables X without highly sensitive variables and X_s represents the input variables X without lowest sensitive variable.

Table 4.2 shows HCS model trained by different combination of input variables.

Table 4.2: **Input vector for each HCS model**

Experiment	Sample data	Dimension	Input variables
1	1	10	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}\ \ddot{\beta_a}$
2	2	10	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}\ \ddot{\beta_a}$
3	3	10	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}\ \ddot{\beta_a}$
4	4	10	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}\ \ddot{\beta_a}$
5	5	10	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}\ \ddot{\beta_a}$
6	6	10	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}\ \ddot{\beta_a}$
7	7	10	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}\ \ddot{\beta_a}$
8	8	10	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}\ \ddot{\beta_a}$
9	1	8	$\beta\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}\ \ddot{\beta_a}$
10	2	8	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}$
11	3	8	$\beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}$
12	4	8	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}$
13	5	9	$\beta\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}\ \ddot{\beta_a}$
14	6	9	$\beta\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}\ \ddot{\beta_a}$
15	7	7	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \ddot{\alpha}\ \ddot{\gamma}$
16	8	7	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\gamma}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}$
17	1	9	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\gamma}\ \ddot{\beta_a}$
18	2	9	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}\ \ddot{\beta_a}$
19	3	9	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\beta_a}$
20	4	9	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\beta_a}$
21	5	9	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}$
22	6	9	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\gamma}\ \ddot{\beta_a}$
23	7	9	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\gamma}\ \ddot{\beta_a}$
24	8	9	$\beta\ \beta_a\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}\ \dot{\beta_a}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}$

In the first 8 experiments, models were trained by input variables X in each human tilt-up skill data. In the experiments 9-16, models were trained by input variables without highly sensitive variables X_l in each set data. Then, in the experiments 17-24, models were trained by input variables without lowest sensitive variable X_s in each set data. Note that in sample data 5 and 6, we could not find extremely high sensitive variable by sensitive analysis. Moreover, the different between highest sensitive variable and lowest sensitive variable are small in these samples. Figure 4.3 and 4.4 illustrate the performance of learned models trained by input state vectors X,X_l and X_s. For details, table 4.3 and 4.4 show the result of learned models.

In the figure 4.3, it shows e_{rms} error of learned models trained by different input vectors X, X_l and X_s. We found that the learned model trained by input vector X had the minimum error. It demonstrated that the input selection in section 4.4.2 was effective. Also, we found that the e_{rms} of the learned models trained by X_l slightly increased while e_{rms} of the learned models trained by X_s increased significantly

except in sample data 5,6. It implied that the networks were difficult to extract the relationship between those sensitive variable to output commands and thus, these variables were unimportant for modeling. On the other hand, the lowest sensitive variables were important for the modeling as the model performance degraded significantly when the input without lowest sensitive variable. In the sample 5 and 6, as we have mentioned before, the performance of learned models were different from the others. Since no extremely highly sensitive value was found, the performance of the models trained by X_l were similar to that of the models trained by X_s. In other words, all state variables in the sample 5 and 6 were important for the modeling.

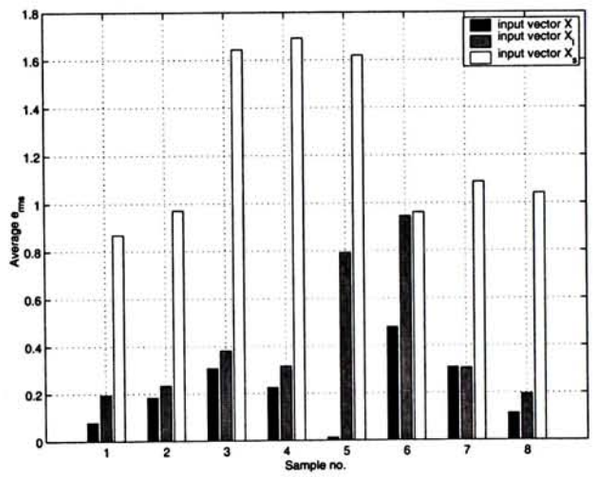

Figure 4.3: **Comparison of learned model trained with input vectors X, X_l and X_s**

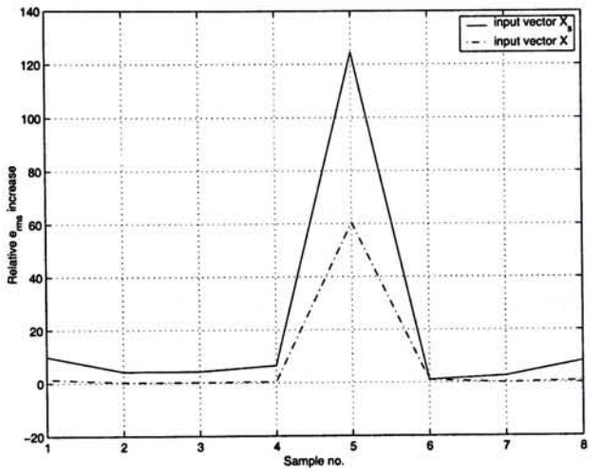

Figure 4.4: **Relative error increase of learned model trained with input vectors X_l and X_s**

Table 4.3: **Performance of learned model trained with input vector X**

Experiment	Sample	$\overline{e_{rms}}$	$\sigma(\overline{e_{rms}})^b$
1	1	0.0794	5.6700×10^{-5}
2	2	0.1841	7.2511×10^{-4}
3	3	0.3062	4.9649×10^{-6}
4	4	0.2254	2.7000×10^{-6}
5	5	0.0129	4.9660×10^{-4}
6	6	0.4795	2.4021×10^{-3}
7	7	0.3085	5.5670×10^{-4}
8	8	0.1133	2.2432×10^{-4}

a. Average root-mean-squared (RMS) error e_{rms} for 15-hidden-unit network (over 30 trials)

b. Standard deviation of the average e_{rms} value for 15-hidden-unit network (over 30 trials)

Table 4.4: **Performance of learned models trained with X_l and X_s**

	Experiment	Sample	$\overline{e_{rms}}$	$\sigma(\overline{e_{rms}})^b$	E^c
X_l	9	1	0.1947	1.1976×10^{-4}	1.45
	10	2	0.2343	1.3885×10^{-4}	0.27
	11	3	0.3809	6.8270×10^{-4}	0.24
	12	4	0.3153	1.1000×10^{-6}	0.39
	13	5	0.7923	4.7648×10^{-5}	60.4
	14	6	0.9459	1.0198×10^{-2}	0.92
	15	7	0.3056	1.2806×10^{-4}	-0.009
	16	8	0.1944	2.3893×10^{-4}	0.715
X_s	17	1	0.8692	1.5620×10^{-3}	9.94
	18	2	0.9708	1.1852×10^{-4}	4.27
	19	3	1.6436	3.2130×10^{-5}	4.36
	20	4	1.6923	3.4908×10^{-4}	6.5
	21	5	1.6192	7.6557×10^{-3}	124
	22	6	0.9618	9.2195×10^{-3}	1.0
	23	7	1.0893	2.8467×10^{-4}	2.5
	24	8	1.0418	4.0624×10^{-4}	8.19

a. Average root-mean-squared (RMS) error e_{rms} for 15-hidden-unit network (over 30 trials)

b. Standard deviation of the average e_{rms} value for 15-hidden-unit network (over 30 trials)

c. Relative e_{rms} increasing$E = \frac{new\ e_{rms} - old\ e_{rms}}{old\ e_{rms}}$ from X to X_l and X_s

4.5 Model-free Method

In the last section, we selected input variables based on the mathematical model of the robot. However, one may argue that the method provides an alternative solution for control problem only and it depends on the existence of the mathematical model. Here, we want to develop model-free approach which does not depend on prior knowledge of system model.

In this section, we will introduce a statistical method, factor analysis, to solve the selection problem. By using factor analysis, sensor variables can be classified into different groups. The variables in intra-group have high correlations among themselves while the variables in inter-group have low correlation with each other. Then, we will select the groups of variables which have high correlation with the control strategy as the model input. The advantage of this method is that the dimension of input vector can be reduced. Moreover, prior knowledge of system model is unnecessary. By this feature, we can further apply this method to any system with unknown model for analyzing and selecting input variables.

4.5.1 Problems Definition

In the previous section, the input selection method depended on the dynamic model of the robot to validate the injective mapping between state space to control command space. Moreover, the last method exists some limitations on the following aspects:

- Prior knowledge of the system is assumed to be known. The situation become critical if the mathematical model of the system is being developed and we cannot select suitable input state variables from the dynamic model of the system. As a result, irrelevant input variables are used to train HCS model.

- Noisy data affect the result of sensitivity analysis. For example, the accelerometer acquires high noise acceleration data. And the sensitivity analysis becomes meaningless under this situation.

Here, we will demonstrate some examples that the performance of HCS models trained by irrelevant input variables. Then, we will show that the sensitivity analysis is weak to handle high noise data.

Influence of Irrelevant Input

Suppose that there do not exist the mathematical model of the robot and we develop different HCS models trained by different inputs variables. Different combination of input variables could affect the performance of learned models. Here, we did two evaluations (total 17 experiments) to examine the effect of different input state variables. We first selected all sensor variables (total 10 sensor variables) as the input vector. They were $X = [\dot{\beta}_a, \beta_a, \dot{\alpha}, \dot{\gamma}, \dot{\beta}, \ddot{x}, \ddot{y}, \ddot{z}, \theta, \beta]^T$. All sensor variables were selected because we wanted to demonstrate the effect of irrelevant input on the modeling. The definition of these notions are shown in table 2.1. Then, we used different combination of input state variables to train HCS models. e_{rms} error is used to evaluate the performance of the learned models.

In the experiment 1 (in table 4.5), all sensor variables were selected as the input vector for the modeling [2]. Then, in the first evaluation (experiment 2-11), we eliminated one state variable in the input vector X and used it to train different HCS models. In the second evaluation(experiment 12-17), we eliminated sensor variables one by one in the input vector in order to demonstrate the effect of different dimension of input vector. The input vector and performance of the learned models are shown in the table 4.5 and figure 4.5 . From these experiments, the performance of HCS models varied with different dimension of input vector. Figure 4.5 shows the e_{rms} of learned models in table 4.5. The performance of the learned models varied sharply and the performance of the learned model upgraded significantly by suitable combination and dimension of input vector. On the other hand, the performance degraded significantly with unsuitable combination of input variables. A HCS model trained by irrelevant input is shown in figure 4.10.

[2] see section 3.4

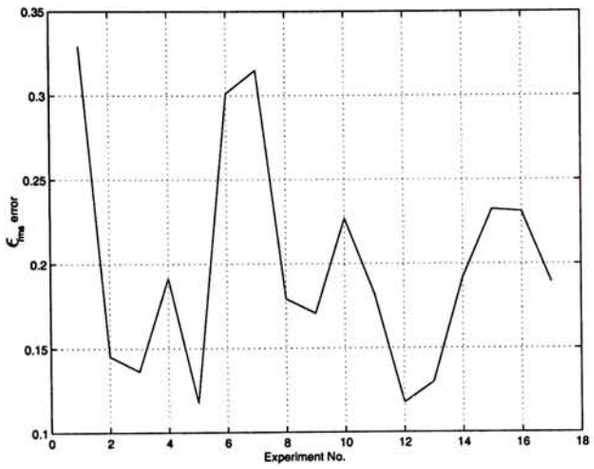

Figure 4.5: **RMS error of HCS models trained with different input vector**

Influence of Noisy Data

In the last section, sensitivity analysis was applied to detect unimportant variables and reduce the dimension of model input. However, that method could not provide a good analysis on high noisy data. To illustrate this problem, we consider one example. Suppose $y(t)$ and $u(t)$ represent the system state and input command respectively. $u(t)$ is the human control strategy u_0 in the figure 2.7 and system state function is defined as equation 4.8.

$$y(t) = \sin(\pi t) + 2\sin(2.5\pi t) \tag{4.8}$$

$$y_1(t) = y(t) + \delta(t) \tag{4.9}$$

$$y_2(t) = y(t) + 10\delta(t) \tag{4.10}$$

where δ is random value. In order to model the sensor data. we add noise on the original system state function $y(t)$ in the equation 4.8. The equation 4.9 and 4.10 shows the result noisy functions and figure 4.6 (a), (b), (c) shows the equation 4.8-4.10. The results of sensitivity analysis are much different from the original one. Therefore, it induces that the sensitivity analysis can not handle the noisy data properly. As a result, we need another evaluation method to select important state variables for the modeling.

Table 4.5: **Performance of HCS models trained by different input vector**

Experiment	N^a	S^b	n_h^c	e_{rms}^d	$\sigma(e_{rms})$
1	10	nil	11	0.3294	0.0135
2	9	$\dot{\beta_a}$	9	0.1454	0.0010
3	9	β_a	7	0.1364	0.0013
4	9	$\dot{\beta}$	8	0.1916	0.0038
5	9	$\dot{\gamma}$	3	0.1177	0.0005
6	9	$\dot{\alpha}$	6	0.3012	0.0263
7	9	$\ddot{x}$	12	0.3150	0.0518
8	9	$\ddot{y}$	14	0.1793	0.0034
9	9	$\ddot{z}$	14	0.1706	0.0040
9	9	θ	4	0.2268	0.0078
11	9	β	13	0.1823	0.0020
12	9	$\dot{\gamma}$	3	0.1177	0.0005
13	8	$\dot{\gamma}, \dot{\alpha}$	5	0.1299	0.0011
14	7	$\dot{\gamma}, \dot{\beta}, \dot{\alpha}$	8	0.1916	0.0030
15	6	$\dot{\gamma}, \dot{\beta}, \dot{\alpha}, \ddot{x}$	3	0.2323	0.0136
16	5	$\dot{\gamma}, \dot{\beta}, \dot{\alpha}, \ddot{x}, \theta$	6	0.2310	0.0193
17	4	$\dot{\gamma}, \dot{\beta}, \dot{\alpha}, \ddot{x}, \ddot{y}, \theta$	7	0.1891	0.0077

a. Number of selected variables

b. Variables are eliminated from the input variable $X = [\dot{\beta_a}, \beta_a, \dot{\alpha}, \dot{\gamma}, \dot{\beta}, \ddot{x}, \ddot{y}, \ddot{z}, \theta, \beta]^T$

c. Number of hidden nodes

d. the average e_{rms} value

4.5.2 Factor Analysis

Here, we want to develop a systematic approach for input selection. By this approach, prior knowledge of the system is not necessary. In order to examine this approach, we assume that we do not know any information about the input-output relationship. We only know that several sensors have been installed on the robot and we can collect these sensor data as the input variables. Therefore, we can only use these variables as the input variables in this section.

Factor analysis is a branch of statistics whose primary purpose is data reduction and summarization. Generally speaking, factor analysis concerns on the problem of analyzing the interrelationships among a large number of variables and then explaining these variables in terms of their underlying dimensions of factors.

Factor analysis is usually concerned with two major problems: (1) reducing the dimensionality of the original data space, whether by principal components or

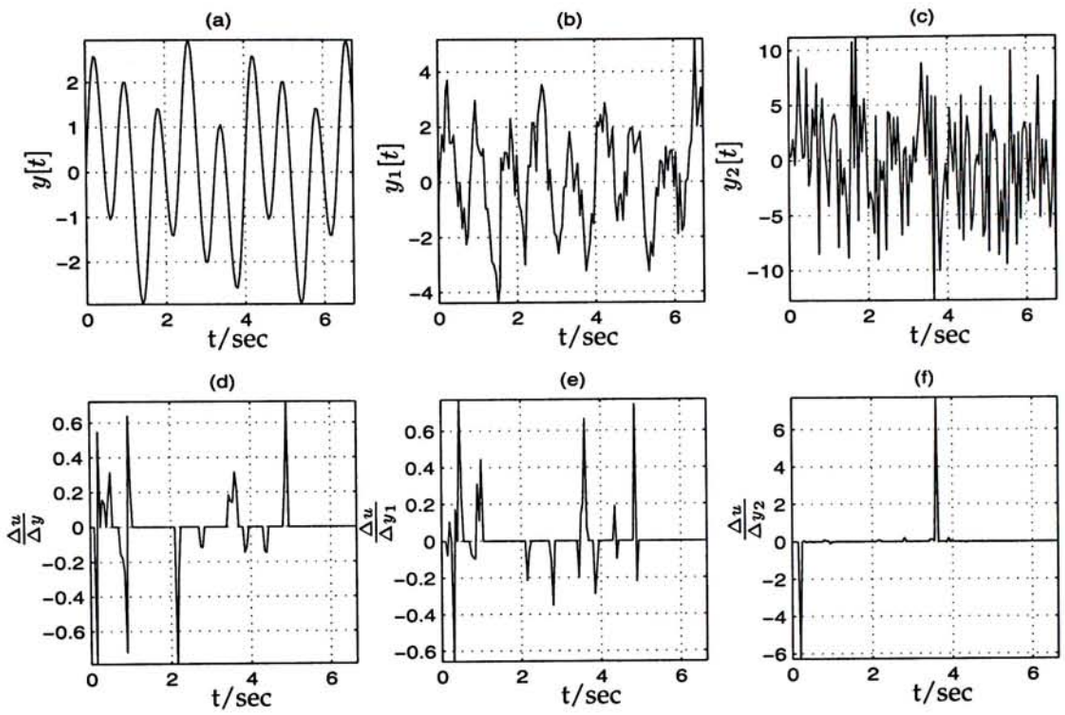

Figure 4.6: **Sensitivity analysis affected by the different degree of noise**

some other factor procedure, and (2) rotation of the factor loading solution in the reduced space to some more interpretable orientation and re-computation of factor scores in the new orientation.

Factor analysis usually operates in a common factor model which assumes a linear relationship from latent factors to observed variables

$$\begin{aligned} y_1 &= \lambda_{11}\xi_1 + \lambda_{12}\xi_2 + \cdots + \lambda_{1r}\xi_r + \epsilon_1 \\ y_2 &= \lambda_{21}\xi_1 + \lambda_{22}\xi_2 + \cdots + \lambda_{2r}\xi_r + \epsilon_2 \\ &\cdots \\ y_m &= \lambda_{m1}\xi_1 + \lambda_{m2}\xi_2 + \cdots + \lambda_{mr}\xi_r + \epsilon_r \end{aligned}$$

or in compact form described as,

$$Y = \Lambda\Xi + \epsilon \tag{4.11}$$

where $Y = [y_1, y_2, \ldots, y_m]^T$ are observed variables measured in derivation from

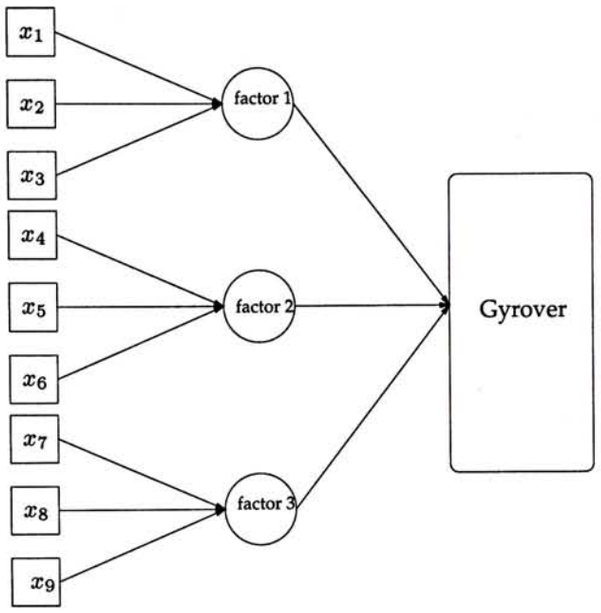

Figure 4.7: **Structural diagram of common factor model on the robot.**

the mean, $\Xi = [\xi_1, \xi_2, \ldots, \xi_r]^T$ are scores for latent factors deviated from the mean, $\Lambda = \lambda_{ij}$ is the factor loading matrix and $\epsilon = [\epsilon_1, \epsilon_2, \ldots, \epsilon_r]^T$ are unique factor scores for observed variables. Usually, random measurement error (sensor noise), influences only a particular observed variable and all other sources of error and bias that prevent the common factors from completely explaining contribute to unique factor ϵ of these observed variables in equation 4.11. The unique factors ϵ are assumed to be uncorrelated with each other and latent factors Ξ. Note that the (i, j) th element λ_{ij} of factor loading matrix Λ reflects the weight for the ith observed variable on the j th factors. Under the common factor model (equation 4.11), the population covariance matrix Σ of Y can be given as,

$$\Sigma = \Lambda \Phi \Lambda^T + \Theta \tag{4.12}$$

where Φ and Θ are covariance matrices of Ξ and ϵ respectively. Θ is a diagonal matrix as unique factors are mutually uncorrelated. The latent factors are not unique defined since any nonsingular transformation matrix A can be applied to Ξ so that the same covariance matrix Σ is reconstructed. As a result, each observed variables depends on some more fundamental set of factor.

Factors Extraction

The main task of factors extraction is to estimate the factor loading matrix Λ, latent factors factors covariance matrix Φ and unique factor covariance matrix Θ under different criteria so that the covariance among observed variables Y are reproduced as possible. Under different factors extraction criteria and assumptions, different factor extraction methods can be derived.

The Unrestricted Maximum Likelihood Estimation method [25] is widely adopted method for factors extraction because it does not impose unrealistic assumptions on unique factors (residual error) structure and it provides tests of statistical significance of parameters estimated. Moreover, maximum likelihood estimator for factor analysis is efficient and consistent. The population of data samples is assumed to be multivariate normal. The maximum likelihood problem can be transformed equivalently to minimize

$$\mathcal{F} = log|\Sigma| + tr[S\Sigma^{-1}] - log|S| - m \tag{4.13}$$

where m is the number of observed variables and S is the sample covariance matrix. In our work, maximum likelihood factor estimation method is employed for the analysis of input space of the robot.

Varimax-Rotated Factor Loadings Analysis

Since infinite pairs of Λ and Ξ can reproduce the same correlations among observed variables with the same data samples which can fulfill the constraint of equation 4.12, it is desired to estimate sets of factor loading patterns with better interpretation on the factors extraction. Then, factor rotation procedures provide refinements and less restrictions on estimated factor loading distributions in matrix Λ. The extracted factors are interpreted in a different perspectives by factor rotation. Here, an orthogonal rotation for latent factor, names *Varimax* scheme [24]. After the orthogonal rotation, rotated latent factors are uncorrelated with each other and the variances of squared factor loading tend to have either high or low magnitudes

and thus this leads to interpretable factors. As a result, each variable depends on a set of factors. The result of factor analysis on sample 1 data is shown in equation 4.14. From this equation, each variable has linear relationship with a number of factors and depends heavily on particular factor. For example, tilt angle β_a mainly depends on factor 1.

$$\begin{aligned}
\dot{\beta}_a &= 0.423\xi_1 + 0.203\xi_2 + 0.194\xi_3 - 0.035\xi_4 + 0.761\xi_5 + 0.080\xi_6 + \epsilon_{\beta_a} \\
\beta_a &= 0.967\xi_1 + 0.141\xi_2 + 0.008\xi_3 - 0.102\xi_4 + 0.084\xi_5 + 0.086\xi_6 + \epsilon_{\dot{\beta}_a} \\
&\vdots \\
u_1 &= 0.949\xi_1 - 0.012\xi_2 + 0.149\xi_3 + 0.136\xi_4 + 0.197\xi_5 + 0.091\xi_6 + \epsilon_{u_1}
\end{aligned} \tag{4.14}$$

Then we can approximate above equations into below form. We selected the terms of the equations with its coefficient begin greater than 0.5.

$$\begin{aligned}
\beta_a &\approx 0.761\xi_5 + \epsilon_{\beta_a} \\
\dot{\beta}_a &\approx 0.967\xi_1 + \epsilon_{\dot{\beta}_a} \\
&\vdots \\
u_1 &\approx 0.949\xi_1 + \epsilon_{u_1}
\end{aligned} \tag{4.15}$$

The simplified equations show that some factors have strong relationship on particular variables. Consequently, we can classify the variables based on factor loading. The variables which have high factor loading on factor j are assigned to group j. Therefore, each variable is assigned to different groups based on:

$$x_k \in G_j \text{ iff } \lambda_{x_k j} > \zeta \tag{4.16}$$

As a result, the variables are classified into different groups.

$$\begin{aligned}
G_1 &= \{\beta, \ddot{x}, \ldots | \lambda_{\beta 1}, \lambda_{\ddot{x}1} \cdots > \zeta\} \\
&\vdots \\
G_i &= \{u_1, \dot{\beta}_a, \ldots | \lambda_{u_1 i}, \lambda_{\dot{\beta}_a i} \cdots > \zeta\} \\
&\vdots
\end{aligned}$$

G_i denotes the group of variables which have high factor loading λ_{xi} on factor i. ζ is a threshold to classify the group of variables. Here, we assign the ζ equal to 0.5. In each group, variables have high correlation among themselves while variables in different groups have low correlation among themselves. For example, in sample 1 data, tilt angle of the flywheel β_a, lean angle of the wheel β, acceleration in X-direction $\ddot{x}$ and tilt torque u_1 are classified into group 1. The change of tilt angle of the flywheel generates tilt torque while, the $\ddot{x}$ influences the rolling speed of the wheel. By the gyroscopic precession (equation 1.1), larger the rolling speed make larger the tilt torque. the lean angle β is the goal of motion. These variables and tilt-torque command u_1 are selected in the same group is reasonable because these state variables have strong relationship with the tilt-torque command u_1. On the same time, the lean angle β and drive motor u_0 are assigned to the same group by factor analysis. The drive motor u_0 controls the rolling speed of the wheel and it seems that the human operator observes the learn angle β (the task goal) and makes corresponding command u_0 to achieve the goal. Figure 4.8 illustrates the concept of factor analysis. In the figure, variables in different groups are almost orthogonal based on the property of factor analysis.

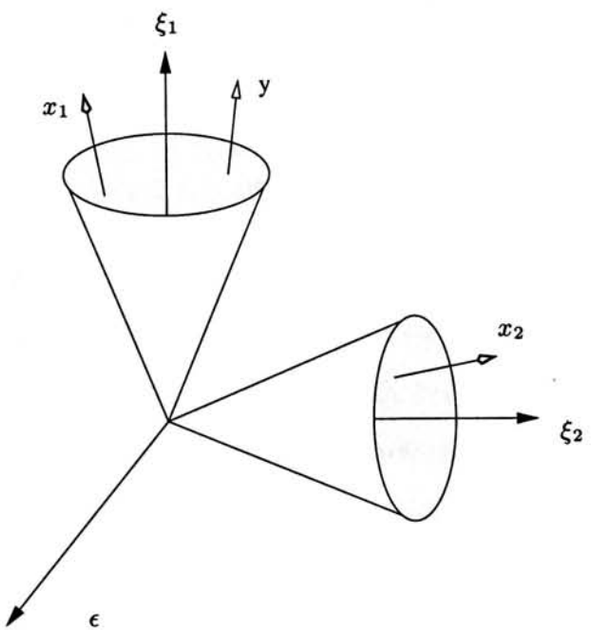

Figure 4.8: **Variables can be classified into different groups based on the magnitude of factor loading**

Table 4.6: **Factor analysis on sample 1**

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Unique Var.
$\dot{\beta_a}$	0.423	0.203	0.194	-0.035	0.761	0.080	0.156
β_a	0.967	0.141	0.008	-0.102	0.084	0.086	0.020
$\dot{\beta}$	0.028	-0.084	0.872	0.123	0.406	0.141	0.031
$\dot{\gamma}$	-0.085	-0.628	-0.007	-0.015	-0.020	0.058	0.594
$\dot{\alpha}$	0.126	0.769	0.217	0.128	0.549	0.152	0.005
$\ddot{x}$	0.651	0.118	0.103	0.326	0.104	-0.171	0.406
$\ddot{y}$	0.117	0.054	0.416	0.118	-0.032	0.826	0.113
$\ddot{z}$	-0.017	-0.034	-0.022	0.070	0.106	0.799	0.344
θ	0.257	0.473	0.787	0.197	-0.110	0.185	0.005
β	0.802	0.205	0.057	0.529	0.111	0.119	0.005
u_0	-0.182	-0.045	-0.160	-0.770	0.014	-0.149	0.324
u_1	0.949	-0.012	0.149	0.136	0.197	0.091	0.011

Input Variable Selection

In the last section, each variable is assigned to different groups by factor analysis. The variables in each group have one common feature that they depend heavily on one latent factor. The variables in same group share partial information and therefore, control command variable u_i can be estimated by the other variables in the same group provided that u_i belongs to this group. For modeling a function from input variables to output command variable, the state variables in the same group of u_i have kept main relevant information with output. Therefore, we define a **Input Selection Rule** such that:

$$\forall G_i, \text{ if } \exists\ u_j \in G_i, \text{ then } \forall x_k \in G_i \text{ are selected as the input state variables} \quad (4.17)$$

By the above algorithm, any variable x_k which has high factor loading on factor i is selected as the input state variable provided that u_j also has high factor loading on factor i. We apply this method to select the variables to form reduced input vector X_r. In the next section, we will compare the performance of reduced input vector with the irrelevant input vector on the modeling

Table 4.7: **Factor analysis on sample 2**

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Unique Var.
$\dot{\beta_a}$	0.643	0.080	0.441	-0.002	0.091	0.128	0.361
β_a	0.819	0.118	-0.013	0.391	0.222	-0.028	0.112
$\dot{\beta}$	0.195	0.003	0.957	-0.081	0.002	0.188	0.005
$\dot{\gamma}$	-0.097	-0.101	0.207	-0.518	-0.091	0.000	0.661
$\dot{\alpha}$	0.208	0.113	0.284	0.185	0.094	0.708	0.319
$\ddot{x}$	0.302	-0.147	0.038	0.156	0.913	0.150	0.005
$\ddot{y}$	0.102	0.717	0.159	0.330	0.056	0.030	0.336
$\ddot{z}$	0.110	0.968	-0.116	-0.049	-0.169	0.033	0.005
θ	0.242	0.093	0.298	0.804	0.072	0.433	0.005
β	0.579	0.137	-0.165	0.138	0.385	0.521	0.179
u_0	0.018	0.045	-0.049	-0.023	-0.046	-0.815	0.329
u_1	0.959	0.086	0.184	0.094	0.112	0.116	0.005

Table 4.8: **Factor analysis on sample 3**

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Unique Var.
$\dot{\beta_a}$	0.544	0.023	-0.028	0.007	0.355	0.582	0.238
β_a	0.962	0.104	0.158	0.149	-0.044	0.096	0.005
$\dot{\beta}$	0.066	0.082	0.005	-0.039	0.864	0.186	0.206
$\dot{\gamma}$	-0.101	-0.022	-0.530	0.017	0.082	-0.167	0.674
$\dot{\alpha}$	0.200	0.001	0.349	-0.195	0.128	0.696	0.300
$\ddot{x}$	0.645	-0.197	0.117	-0.353	-0.001	0.134	0.389
$\ddot{y}$	0.053	0.711	0.188	-0.042	0.279	-0.105	0.366
$\ddot{z}$	0.027	0.984	-0.063	0.013	-0.098	0.111	0.005
θ	0.080	0.137	0.767	-0.226	0.574	-0.043	0.005
β	0.730	0.090	0.178	-0.622	-0.066	0.174	0.005
u_0	-0.031	-0.015	-0.061	0.762	-0.076	-0.066	0.404
u_1	0.911	0.102	0.020	-0.068	0.197	0.135	0.097

Table 4.9: **Factor analysis on sample 4**

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Unique Var.
$\dot{\beta_a}$	0.323	0.146	0.496	0.159	0.651	0.126	0.163
β_a	0.970	0.079	-0.004	-0.148	-0.070	0.092	0.017
$\dot{\beta}$	-0.218	-0.035	0.113	0.013	0.730	-0.022	0.404
$\dot{\gamma}$	0.010	-0.227	-0.311	-0.293	0.514	-0.454	0.295
$\dot{\alpha}$	0.116	0.018	0.958	-0.008	0.163	0.191	0.005
$\ddot{x}$	0.736	0.106	0.206	0.034	-0.044	0.186	0.367
$\ddot{y}$	0.146	0.603	0.118	-0.089	0.035	0.266	0.521
$\ddot{z}$	0.085	0.975	-0.053	-0.095	-0.073	-0.140	0.005
θ	0.337	0.032	0.245	-0.345	0.008	0.768	0.118
β	0.926	0.091	0.175	-0.233	-0.153	0.138	0.007
u_0	-0.344	-0.231	0.034	0.817	0.029	-0.193	0.122
u_1	0.962	0.109	-0.034	-0.176	0.083	0.013	0.024

Table 4.10: **Factor analysis on sample 5**

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Unique Var.
$\dot{\beta_a}$	0.906	0.176	0.086	0.004	0.235	0.283	0.005
β_a	0.056	0.976	-0.025	-0.061	0.077	-0.144	0.014
$\dot{\beta}$	0.359	-0.268	0.073	0.152	0.021	0.875	0.005
$\dot{\gamma}$	0.021	0.010	0.989	-0.013	0.095	0.081	0.005
$\dot{\alpha}$	0.282	0.315	0.169	-0.011	0.887	0.025	0.005
$\ddot{x}$	0.001	0.707	-0.055	0.020	0.172	-0.035	0.465
$\ddot{y}$	-0.045	0.229	0.047	0.965	-0.087	-0.011	0.005
$\ddot{z}$	0.014	-0.070	-0.071	-0.725	0.070	-0.093	0.451
θ	-0.152	0.539	-0.469	-0.254	-0.127	0.087	0.377
β	0.057	0.933	-0.120	-0.062	0.212	-0.188	0.028
u_0	0.517	-0.640	-0.050	0.043	0.148	0.148	0.274
u_1	0.095	0.949	0.106	-0.070	0.085	0.008	0.067

Table 4.11: **Factor analysis on sample 6**

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Unique Var.
$\dot{\beta_a}$	0.301	0.082	-0.017	0.780	-0.001	0.354	0.169
β_a	0.970	0.080	0.177	-0.049	-0.007	0.039	0.017
$\dot{\beta}$	-0.284	-0.049	0.019	0.725	0.234	0.048	0.334
$\dot{\gamma}$	-0.015	-0.077	-0.054	0.153	0.765	-0.054	0.379
$\dot{\alpha}$	0.213	0.031	0.150	0.333	-0.076	0.677	0.357
$\ddot{x}$	0.731	0.034	0.134	0.043	-0.124	0.176	0.399
$\ddot{y}$	0.142	0.551	0.320	0.079	-0.111	0.052	0.552
$\ddot{z}$	0.049	0.994	-0.037	-0.049	-0.010	-0.001	0.005
θ	0.357	0.061	0.839	0.115	-0.378	0.059	0.005
β	0.916	0.102	0.273	-0.080	-0.093	0.207	0.018
u_0	-0.337	-0.180	-0.631	0.107	-0.241	-0.162	0.360
u_1	0.956	0.109	0.189	0.073	0.121	0.014	0.018

Table 4.12: **Input vector for each experiments**

Experiment	Source	Input vector X	Experiment	Source	Input vector X_r
1	1	$\dot{\beta_a}, \beta_a, \dot{\alpha}, \dot{\gamma}, \dot{\beta}, \ddot{x}, \ddot{y}, \ddot{z}, \theta, \beta$	7	1	$\beta_a, \ddot{x}, \beta$
2	2	$\dot{\beta_a}, \beta_a, \dot{\alpha}, \dot{\gamma}, \dot{\beta}, \ddot{x}, \ddot{y}, \ddot{z}, \theta, \beta$	8	2	$\dot{\beta_a}, \beta_a, \dot{\alpha}, \beta$
3	3	$\dot{\beta_a}, \beta_a, \dot{\alpha}, \dot{\gamma}, \dot{\beta}, \ddot{x}, \ddot{y}, \ddot{z}, \theta, \beta$	9	3	$\dot{\beta_a}, \beta_a, \ddot{x}, \beta$
4	4	$\dot{\beta_a}, \beta_a, \dot{\alpha}, \dot{\gamma}, \dot{\beta}, \ddot{x}, \ddot{y}, \ddot{z}, \theta, \beta$	10	4	$\beta_a, \ddot{x}, \beta$
5	5	$\dot{\beta_a}, \beta_a, \dot{\alpha}, \dot{\gamma}, \dot{\beta}, \ddot{x}, \ddot{y}, \ddot{z}, \theta, \beta$	11	5	$\dot{\beta_a}, \beta_a, \ddot{x}, \theta, \beta$
6	6	$\dot{\beta_a}, \beta_a, \dot{\alpha}, \dot{\gamma}, \dot{\beta}, \ddot{x}, \ddot{y}, \ddot{z}, \theta, \beta$	12	6	$\beta_a, \ddot{x}, \beta$

4.5.3 Experimental Result

Model Training

In this section, we apply the input selection rule to select input variables. Table 4.6-4.11 show six sets of training data for factor analysis. Small frame boxes in the tables highlight that variable in this row has high factor loading on the factor on this column. The state variables in the tables are select as input vector if they meet the requirement of the input selection rule(factor loading > 0.5). In order to make fair comparison of input vectors, we compared the models trained by irrelevant input vector $\{\dot{\beta_a}, \beta_a, \dot{\alpha}, \dot{\gamma}, \dot{\beta}, \ddot{x}, \ddot{y}, \ddot{z}, \theta, \beta\}$ (we have mentioned in section 4.5.1) and and reduced input vector in each training data. We note that notion X represents the irrelevant input vector and X_r represents the reduced input vector in later description. In the table 4.6 , we found that $\beta_a, \ddot{x}, \beta, u_1$ had high factor loading on factor 1. And β, u_0 belonged to factor 4. Therefore, we selected $\beta_a, \ddot{x}, \beta$ as the input vector for sample data 1. By similarity , we selected $\dot{\beta_a}, \beta_a, \dot{\alpha}, \beta$ as the input vector in the table 4.7. In the table 4.8, $\dot{\beta_a}, \beta_a, \ddot{x}, \beta$ were selected as the input vector for sample data 3. In the table 4.9, $\beta_a, \ddot{x}, \beta$ were selected as the input vector for sample data 4. In the table 4.10, $\dot{\beta_a}, \beta_a, \ddot{x}, \theta, \beta$ are selected as the input vector for sample data 5. In the table 4.11, we selected $\beta_a, \ddot{x}, \beta$ as the input vector for sample data 6. Above selection of input vectors are shown in table 4.12.

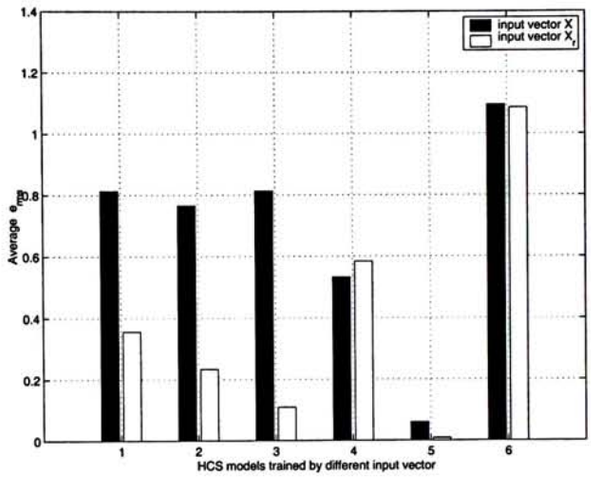

Figure 4.9: **Comparison of the learned model trained with input vector X and reduce vector X_r**

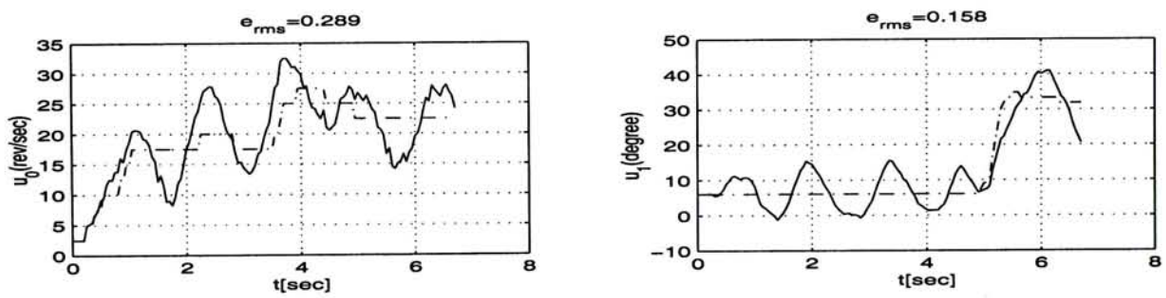

Figure 4.10: **sample 1: learned model trained with input vector X**

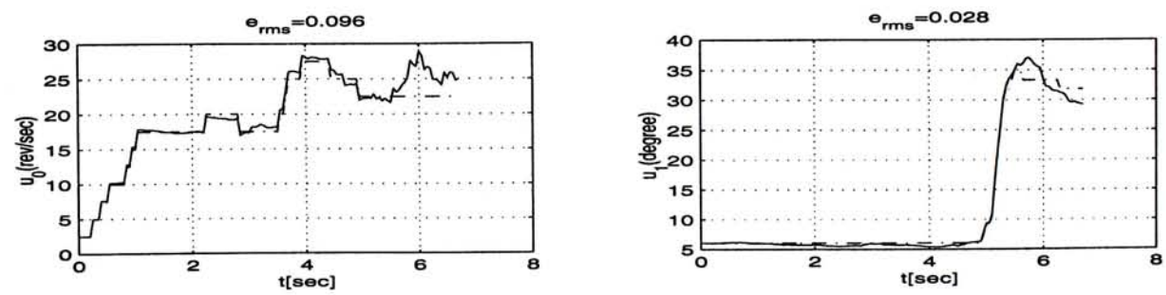

Figure 4.11: **sample 1: learned model trained with reduced input vector X_r**

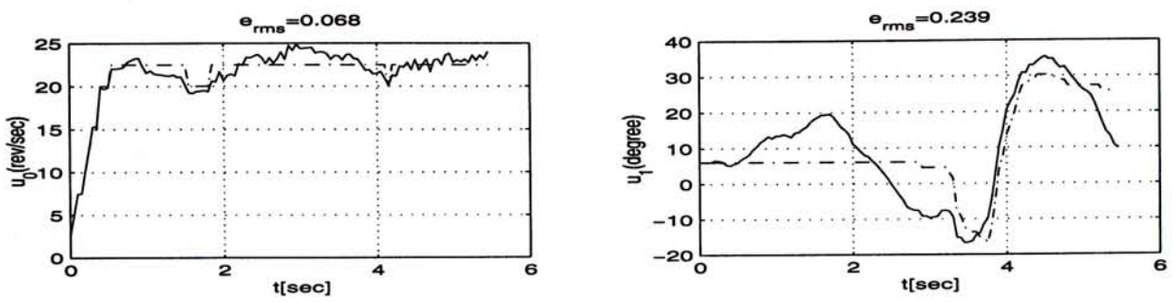

Figure 4.12: **sample 3: learned model trained with input vector X**

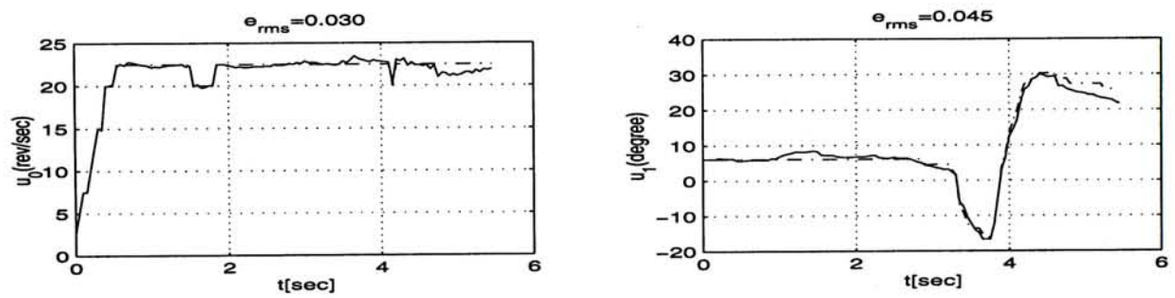

Figure 4.13: **sample 3: learned model trained with reduced input vector X_r**

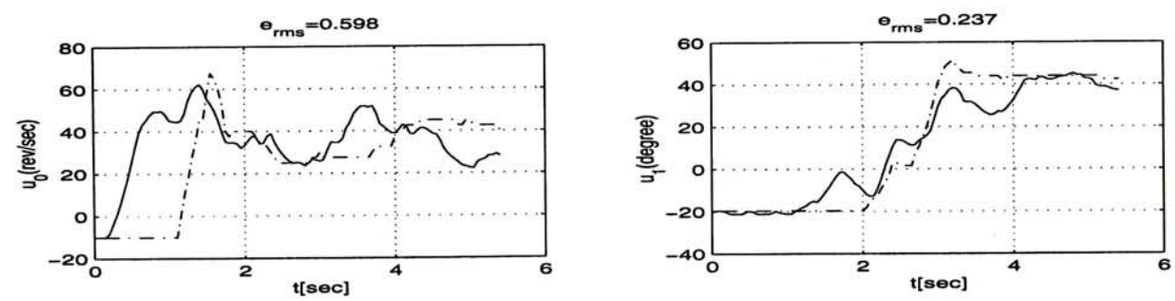

Figure 4.14: **sample 6: learned model trained with input vector X**

Analysis

Tables 4.13 and 4.14 illustrate the experimental results of HCS model trained by the irrelevant input vector and the reduced input vector respectively. For convenience, $RMS(X)$ denotes the average e_{rms} error of learned model output (compared with source human strategy). In the first three sets of data, the $RMS(X_r)$s decrease significantly, compared with $RMS(X)$. For example, in the sample 1, the $RMS(X_r)$ decreases 41 percentage of the $RMS(X)$. In the second three set of

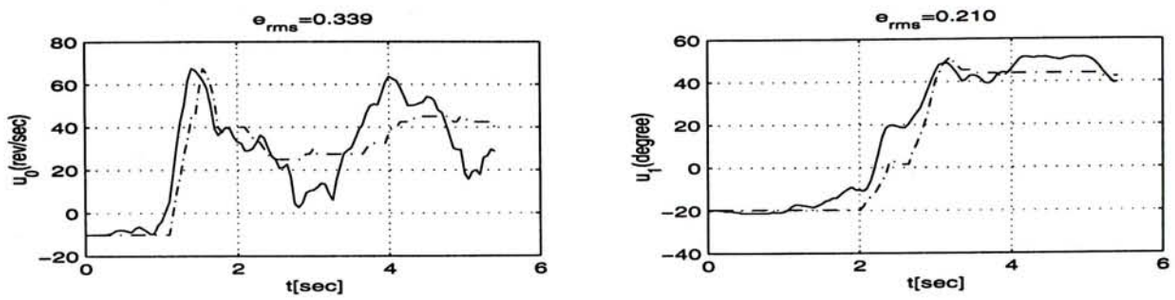

Figure 4.15: **sample 6: learned model trained with reduced input vector X_r**

data, the $RMS(X_r)$s are almost the same as $RMS(X)$. From above results, we can conclude that the average e_{rms} error of the models trained with the reduced input vector X_r are better than that of the original input vector X. Moreover, the dimension of input vector can be reduced by this selection method. For example, in the sample 1, the dimension of the reduced input vector X_r is only 3 while the dimension of the input vector X is 10.

It is interesting to note that the $RMS(X_r)$ is slightly higher than $RMS(X)$ in the sample 4 (see experiment 4 and 10). From the table 4.9, we could not find any state variable related to the control command u_0 by factor analysis. The reduced input vector might miss important information for the modeling. Consequently, the learned model trained by reduced input vector has poor performance relative to the learned model trained by irrelevant input vector.

Figure 4.10-4.15 show some representative learned models result. Figures 4.11, 4.13,4.15 illustrate the learned strategies of the model trained by X_r. The dotted line shows the source human control strategies and the solid line shows the strategies of the learned models. Meanwhile, figures 4.10,4.12,4.14 illustrate the learned strategies of the model trained by X.

4.6 Model-based Method versus Model-free Method

Here, we compare two selection methods for the modeling. In the two methods, we used different combination of input variables to train different HCS models and each models has different performances. In order to standardize comparison,

Table 4.13: **Performance of learned model trained the input vector X**

Experiment	Sample	$\overline{e_{rms}}^b$	$\sigma(\overline{e_{rms}})^c$
1	1	0.8130	0.0167
2	2	0.7650	0.0061
3	3	0.8130	0.0167
4	4	0.5326	0.0051
5	5	0.0596	0.0132
6	6	1.0953	0.0609

a. Variables are eliminated from the input vector X

b. Average root-mean-squared (RMS) error e_{rms} for 15-hidden-unit network (over 30 trials)

c. Standard deviation of the average e_{rms} value for 15-hidden-unit network (over 30 trials)

Table 4.14: **Performance of learned model trained the input vector X_r**

Experiment	Sample	$\overline{e_{rms}}^a$	$\sigma(\overline{e_{rms}})^b$	E^c
7	1	0.3566	5.0000×10^{-4}	-0.56
8	2	0.2341	6.8505×10^{-4}	-0.69
9	3	0.1095	1.7557×10^{-4}	-0.86
10	4	0.5837	5.9816×10^{-3}	0.09
11	5	0.0079	3.1294×10^{-5}	-0.86
12	6	1.0853	1.5402×10^{-2}	-0.009

a. Average root-mean-squared (RMS) error e_{rms} for 15-hidden-unit network (over 30 trials)

b. Standard deviation of the average e_{rms} value for 15-hidden-unit) network (over 30 trials)

c. Relative e_{rms} increasing$E = \frac{new\ e_{rms} - old\ e_{rms}}{old\ e_{rms}}$ from X to X_r

we only use the learned models showed in table 4.3 (the model-approach method) and table 4.13 (the model-free method) because the learned models in these tables have best performance in each method. Thus, the best learned models trained by two methods will be compared in this section.

The first six sets of the learned models in the table 4.3 are trained by sample data 1-6. Also, the six sets of the learned models in the table 4.13 are trained by sample data 1-6. Note that table 4.3 shows eight sets of data while table 4.13 only shows six sets of data. Because the last two sets of training data (sample data 7-8) cannot find any input variables by factor analysis. This is one of weakness of factor analysis that sample data being factor analysis may have possibility that the data cannot be factorized.

Figure 4.16 compares e_{rms} of the learned models trained by the first method

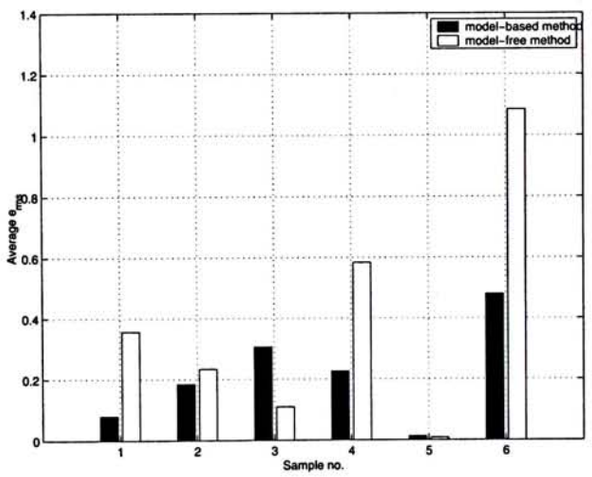

Figure 4.16: e_{rms} **of learned models trained by the model-based method and the model-free method**

and the second method. From the figure, we find that four sets of learned models trained by the model-based method are less error than those models trained by the model-free method and we conclude that the learned model trained by the model-based method have better performance than the learned model trained by the model-free method.

Table 4.15 shows the comparison between the model-based method and the model-free method. The model-free method is lowly sensitive to noise data since

Table 4.15: **Model-based method vs model-free method**

Category	Model-based method	Model-free method
Prior knowledge	high	low
Sensitive to noise	high	low
Performance	high	medium
Input dimension	high	low

because this method already considers the noise data during the statistical method, factor analysis. In our cases, the input variables dimension of the model-based method are kept in 10 while the input variables dimension of the model-free method are always less than 5.

The requirement of the model-based method is higher than that of the model-free method. The performance of the model-based method is higher, however. On the other hand, the performance of the model-free method do not need the prior knowledge of the system and is lowly sensitive to noise data. Also, the input variables dimension is always smaller than that of the model-based method. We can use the two methods based on different situation.

In conclusion, we developed two evaluation methods to select suitable input variables to handle the selection problem in different situations. In the first evaluation method, we first selected the relevant state variables from the mathematical model of the robot as the HCS model input. Then, we defined a measure of sensitivity of each state variables with respect to the operator's control input to validate the importance of model input. In the second evaluation method, we presented a statistical method, *factor analysis* , to select the suitable state variables. By using factor analysis, the variables could be classified into different groups. Intra-group variables had high correlations among themselves while inter-group variables had low correlation among themselves. Then, we selected the groups of variables which had high correlation with the human strategies as the input variables. The advantage of this evaluation method was that prior knowledge of the mathematical model of the system was unnecessary. Besides, the overall performance of HCS models upgraded after the reduction of model input by eliminating uncorrelated input. That

method could be further applied to any systems without any model for analyzing and selecting input variables.

Chapter 5

Conclusion and Future Work

5.1 Contributions

In this thesis, we presented a study of methodology for selecting input variables and modeling human control strategy (HCS) on the single wheel robot. We also practically implemented HCS model to control the "tilt-up" motion of the robot. We summarize the original contributions of this work below:

- **Human-based Control** We applied flexible cascade neural network with node-decoupled extended Kalman filtering to abstract human control skill on the single wheel robot. And we transferred the model effectively in controlling the robot. We experimentally validated that the HCS model could abstract human skills and the human-based controller could control the robot.

- **Input selection for the modeling** We developed two input selection method to select suitable input variables. One was called model-based method and the other was called model-free method. In the first method, we first identified the effective state variables as the input variables from the mathematical model. Then, we defined a method, sensitivity analysis, to verify the importance of these variables. By this method, any unimportant variable can be eliminated in the bracket of input variables. We used different input combination to train HCS models and compared the corresponding model error to verify the selection method. In the second method, we applied a statistical

method, factor analysis, to classify the state variables into different groups. The advantage of the method was that prior knowledge of the system model was unnecessary. Moreover, the input dimension could be further reduced. By that feature, we could further apply this method to a system with unknown model for analyzing and selecting input variables.

5.2 Future Work

In this thesis, we provide the foundation of modeling HCS and its input selection to the robot. It is only the first step toward the automatic control of the robot by human-based controller. The following are possible improvement and extensions of this work:

- **Skill evaluation** Once we have abstracted a HCS model, it is important to access the skill exhibited by the model and its corresponding human controller. In this thesis, we evaluate model based on e_{rms} error. There are, however, other criteria– many of them task-dependent—by which we can evaluate performance of models. Models or control strategies with different skill qualities may be more or less appropriate for a given situation, depending on the specific performance requirement of the robot. Some related research have worked on this area, Xu and Song [8] have studied the issue of skill evaluation by proposing two task-specific performance criteria for the human driving task.

- **Model optimization** Given a specific requirement, it might be necessary to optimize a particular HCS model with respect to that performance measure. The unoptimized HCS model already gives an initial control strategy; optimization would refine the parameters in the model to improve performance with respect to a specific criterion.

- **Automatic Control** Up to now, we have only modeled tilt-up skill of the single wheel robot. Our goal is to automatic control of the robot by human-based

controller. Therefore, we need to model different skill in controlling the robot and integrate the models together.

Appendix A

Dynamic Model of the Robot

A.1 Kinematic Constraints: Holonomic and Nonholonomic

Gyrover has been controlled only manually, using two joysticks to control the drive and tilt motors through a radio link. In the following sections, we will describe the works done by done by Au and Xu[2] on the nonholonomic kinematics constraints, as well as dynamic model using the constrained generalized Lagrangian formulation.

A.1.1 Coordinate Frame

The equations of motion of the robot is assumed that the wheel is a rigid, homogeneous disk which rolls over a perfectly flat surface without slipping. The actuation mechanism, suspended from the wheel bearing, as a two-link manipulator, with a spinning disk attached at the end of the second link (Figure 2.1) are modeled. The first link of length l_1 represents the vertical offset of the actuation mechanism from the axis of the Gyrover wheel. The second link of length l_2 represents the horizontal offset of the spinning flywheel and is relatively smaller compared to the vertical offset.

Next, four coordinates frames are defined as follows: (1) the inertial frame $\sum_O$,

whose $x-y$ plane is anchored to the flat surface, (2) the body coordinate frame $\sum_B$ $\{x_B, y_B, z_B\}$, whose origin is located at the center of the single wheel, and whose z-axis represents the axis of rotation of the wheel, (3) the coordinate frame of internal mechanism $\sum_C$ $\{x_c, y_c, z_c\}$, whose center is located at point D, and whose z-axis is always parallel to z_B, and (4) the flywheel coordinates frame $\sum_E$ $\{x_a, y_a, z_a\}$, whose center is located at the center of the Gyrover flywheel, and whose z-axis represents the axis of rotation of the flywheel. Note that y_a is always parallel to y_c. The definition and configuration of system and variables are shown in Table **??** and Figure 2.1. Rolling without slipping is a typical example of a nonholonomic system, since in most cases, some of the constrained equations for the system are nonintegrable. Gyrover is a similar type of the nonholonomic system. (i, j, k) and (l, m, n) are defined to be the unit vectors of the coordinate system $XYZO(\sum_O)$ and $x_B y_B z_B A(\sum_B)$, respectively. Let $S_x := sin(x)$ and $C_x := cos(x)$. The transformation between these two coordinate frames is given by

$$\begin{bmatrix} i \\ j \\ k \end{bmatrix} = R_B^O \begin{bmatrix} l \\ m \\ n \end{bmatrix} \tag{A.1}$$

where R_B^O is the rotation matrix from $\sum_O$ to $\sum_B$.

$$R_B^O = \begin{bmatrix} -S_\alpha C_\beta & -C_\alpha & -S_\alpha S_\beta \\ C_\alpha C_\beta & -S_\alpha & C_\alpha S_\beta \\ -S_\beta & 0 & C_\beta \end{bmatrix} \tag{A.2}$$

Let v_A and ω_B denote the velocity of the center of mass of the single wheel and its angular velocity with respect to the inertia frame $\sum_O$. Then, we have

$$\omega_B = -\dot{\alpha} S_\beta l + \dot{\beta} m + (\dot{\gamma} + \dot{\alpha} C_\beta) n \tag{A.3}$$

The constraints require that the disk rolls without slipping on the horizontal plane, i.e, the velocity of the contact point on the disk is zero at any instant

$$v_c = 0, \tag{A.4}$$

where v_c is the velocity of contact point of the single wheel. Now, we can express v_A as

$$v_A = \omega_B \times r_{AC} + v_c \tag{A.5}$$

where $r_{AC} = -Rl$ representing the vector from the frame C to A in Figure 5. Substituting Eqs. (A.3) and (A.4) in Eq. (A.5) gives

$$v_A = \dot{X}i + \dot{Y}j + \dot{Z}k, \tag{A.6}$$

where

$$\dot{X} = R(\dot{\gamma}C_\alpha + \dot{\alpha}C_\alpha C_\beta - \dot{\beta}S_\alpha S_\beta) \tag{A.7}$$

$$\dot{Y} = R(\dot{\gamma}S_\alpha + \dot{\alpha}C_\beta S_\alpha + \dot{\beta}C_\alpha S_\beta) \tag{A.8}$$

$$\dot{Z} = R\dot{\beta}C_\beta \tag{A.9}$$

Eqs. (A.7) and (A.8) are nonintegrable and hence are nonholonomic while Eq. (A.9) is integrable, i.e,

$$Z = RS_\beta. \tag{A.10}$$

Therefore, the robot can be represented by seven (e.g. $X, Y, \alpha, \beta, \gamma, \beta_a, \theta$), instead of eight, independent variables.

A.2 Robot Dynamics

In this section, the equation of motion is calculated from Lagrangian $L = T - P$ of the system, where T and P are the kinetic energy and potential energy of the system respectively. The system is divided into three parts: 1) single wheel, 2) internal mechanism, 3) spinning flywheel.

A.2.1 Single Wheel

The kinetic energy of the single wheel is given by,

$$\begin{aligned} T_w &= \frac{1}{2}m_w\left[\dot{X}^2+\dot{Y}^2+\dot{Z}^2\right] \\ &\quad +\frac{1}{2}\left[I_{xxw}\omega_x^2+I_{yyw}\omega_y^2+I_{zzw}\omega_z^2\right] \end{aligned} \tag{A.11}$$

Substituting Eqs.(A.3) and (A.9) in Eq.(A.11) yields

$$\begin{aligned} T_w &= \frac{1}{2}m_w\left[\dot{X}^2+\dot{Y}^2+(R\dot{\beta}C_\beta)^2\right]+\frac{1}{2}\left[I_{xxw}(\dot{\alpha}S_\beta)^2\right. \\ &\quad \left.+I_{yyw}\dot{\beta}^2+I_{zzw}(\dot{\alpha}C_\beta+\dot{\gamma})^2\right] \end{aligned} \tag{A.12}$$

The potential energy of the single wheel is

$$P_w = m_w g R S_\beta \tag{A.13}$$

A.2.2 Internal Mechanism and Spinning Flywheel

In the following, the translational and rotational parts of kinetic energy are computed for the internal mechanism and flywheel respectively. l_2 is assumed to be very small compared with l_1,

$$l_2 \simeq 0 \tag{A.14}$$

Thus, the flywheel's center of mass (E) coincides with that of the internal mechanism (D).

Let $\{x_f, y_f, z_f\}$ be the center of mass of the internal mechanism and the flywheel w.r.t. $\sum_O$. The transformation from the center of mass of single wheel to the flywheel can be described:

$$\begin{bmatrix} x_f \\ y_f \\ z_f \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + R_B^O \begin{bmatrix} l_1 C_\theta \\ l_1 S_\theta \\ 0 \end{bmatrix} \tag{A.15}$$

Let T_f^t denote the translational kinetic energy of the flywheel and the internal mechanism.

$$T_f^t = \frac{1}{2}(m_i + m_f)[\dot{x}_f^2 + \dot{y}_f^2 + \dot{z}_f^2] \tag{A.16}$$

Differentiating Eq. (A.15) and substituting it in Eq. (A.16), T_f^t is obtained. Let ω_f be the angular velocity of flywheel w.r.t. $\sum_O$. We then have

$$\omega_f = R_B^E \omega_B + \begin{bmatrix} 0 \\ \dot{\beta}_a \\ \dot{\gamma}_a \end{bmatrix} \tag{A.17}$$

where R_B^E is the transformation from $\sum_B$ to $\sum_E$.

$$R_B^E = \begin{bmatrix} C_\theta S_{\beta_a} & -S_\theta S_{\beta_a} & C_{\beta_a} \\ S_\theta & C_\theta & 0 \\ C_\theta C_{\beta_a} & -C_{\beta a} S_\theta & S_{\beta_a} \end{bmatrix} \tag{A.18}$$

The rotational kinetic energy of the flywheel is now given by,

$$T_f^r = \frac{1}{2}\left[(\omega_{fx})^2 I_{xxf} + (\omega_{fy})^2 I_{yyf} + (\omega_{fz})^2 I_{zzf}\right] \tag{A.19}$$

The flywheel is assumed to be a uniform disk, the principle moments of inertia are $I_{xxf} = I_{yyf} = \frac{1}{4}m_f r^2, I_{zzf} = \frac{1}{2}m_f r^2$. The potential energy of the flywheel and internal mechanism is

$$P_f = (m_i + m_f)(RS_\beta - l_1 C_\theta S_\beta) \tag{A.20}$$

A.2.3 Lagrangians of the System

The Lagrangian of the system thus is

$$L = [T_w + (T_f^t + T_f^r)] - (P_w + P_f) \tag{A.21}$$

Substituting Eqs. (A.11), (A.16), (A.19), (A.13) and (A.20) in Eq. (A.21), we may determine L. There are only two control torques available on the system. One

is drive torque (u_1) and the other is the tilt torque (u_2). Consequently, using the constrained Lagrangian method, the dynamic equation of the entire system is given by

$$M(q)\ddot{q} + N(q,\dot{q}) = A^T\lambda + Bu \tag{A.22}$$

where $M(q) \in R^{7\times7}$ and $N(q,\dot{q}) \in R^{7\times1}$ are the inertia matrix and nonlinear terms respectively.

$$A(q) = \begin{bmatrix} 1 & 0 & -RC_\alpha C_\beta & RS_\alpha S_\beta & -RC_\alpha & 0 & 0 \\ 0 & 1 & -RC_\beta S_\alpha & -RC_\alpha S_\beta & -RS_\alpha & 0 & 0 \end{bmatrix} \tag{A.23}$$

$$q = \begin{bmatrix} X \\ Y \\ \alpha \\ \beta \\ \gamma \\ \beta_a \\ \theta \end{bmatrix}, \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ k_1 & 0 \\ 0 & 1 \\ k_2 & 0 \end{bmatrix}, u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

The nonholonomic constraints can be written as,

$$A(\dot{q})\dot{q} = 0. \tag{A.24}$$

It is noted that *all elements of the last two columns of the matrix A are zero*, because the nonholonomic constraints only restrict the motion of the single wheel, not the flywheel. The last two columns represent the motion variables of the flywheel. Moreover, *the matrix B only have three rows with nonzero elements* since the input torques only drive the tilt angle of the flywheel (β_a) and the rotating angle of the single wheel (γ), so that the fifth and the sixth rows of B are non-zero as they represent the tilting motion of the flywheel and the rotating motion of the single wheel respectively. Furthermore, when the single wheel rotates, the pendulum motion of internal mechanism is introduced, thus θ changes. Therefore, the drive torque of the single wheel will also affect the pendulum motion of the internal mechanism (θ), so that the seventh row of the matrix B is not zero.

Appendix B

Similarity Measure

Similarity measures have been applied on computer vision[43], image database retrieval [44] and 2d shape analysis [45]. However, these methods usually depend on particular properties of images and thus, is not appropriate for analyzing HCS trajectories. One similarity measure for validating human control strategy models has been developed by Nechyba and Xu. This method is based on Hidden Markov Model (HMM) which has been used in many applications, especially a human control data, it can be used to evaluate stochastic similarity between two dynamic multi-dimensional trajectories using HMMs analysis.

A Hidden Markov Model consists of a set of n states, interconnected through probabilistic transitions. Each of these states has some output probability distribution associated with it. A discrete HMM is completely defined by the following triplet [26],

$$\lambda = \{A, B, \pi\} \tag{B.1}$$

where A represents the probabilistic $n \times n$ state transition matrix, B represents the $l \times n$ output probability matrix with l discrete output symbols, and π represents the n-length initial state probability distribution vector.

Denote O as discrete symbol of an observation sequence. The probability of the model λ given for the observation sequence O is defined as $P(\lambda|O)$ and the probability that a given observation sequence O is generated from the model λ is defined by $P(O|\lambda)$. The notion of equivalent HMMs for two HMMs λ_1 and λ_2 such that

$$\lambda_1 \sim \lambda_2, \text{ iff } P(O|\lambda_1) = P(O|\lambda_2), \forall O \tag{B.2}$$

let

$$P_{ij} = P(\bar{O}_i|\lambda_j)^{1/T_i}, \quad i,j \in \{1,2\} \tag{B.3}$$

where the probability fo the observation sequences $\bar{O}_i$ given the model λ_j, normalized with respect to the sequence lengths T_i. Then, the similarity measure is defined as

$$\sigma(\bar{O}_1, \bar{O}_2) = \sqrt{\frac{P_{21}P_{12}}{P_{11}P_{22}}} \tag{B.4}$$

The similarity measure can be used to validate the model-generated trajectories and the HCS. By this method, the best learned-model can be selected which has highest scores of similarity measure for HCS controller.

Bibliography

[1] M.C. Nechyba, *Learning and Validation of Human Control Strategies*, Ph.D. thesis, Carnegie Mellon University, 1998

[2] K.W. Au, *Dynamics and Control of a Single Wheel, Gyroscopically Stabilized Robot*, M.Phil. thesis, Chinese University of Hong Kong, 1999

[3] Y. Xu, W.K. Yu and K.W. Au " Modeling Human Strategy in Control a Dynamically Stabilized Robot", *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1999.

[4] M.C. Nechyba and Y. Xu, "On the fidelity of human skill models," *IEEE Int. Conf. on Robotics and Automation*, Vol. 3, pp. 2688-2693, 1996

[5] J. Yang, Y. Xu and C.S. Chen, "Hidden Markov Model Approach to Skill Learning and its Application to Telerobotics," *IEEE Trans. on Robotic and Automation*, Vol. 10, no. 5 pp. 621-31. 1994

[6] M.C. Nechyba and Y. Xu, "Cascade Neural Network with Node-Decoupled Extended Kalman Filtering", *Proc IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, Vol. 1, pp. 214-9, 1997.

[7] M.C. Nechyba and Y. Xu, "Human Control Strategy: Abstraction, Verification and Replication", *IEEE Control Systems Magazine*, Vol. 17, no. 5, pp. 48-61, 1997.

[8] J.Y. Song, Y. Xu, M.C. Nechyba and Y. Yam, "Two Measures for Evaluating Human Control Strategy", *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1998.

[9] H. B. Brown and Y. Xu, "A Single Wheel Gyroscopically Stabilized Robot." *Proc. IEEE Int. Conf. on Robotic and Automation*, Vol. 4, pp. 3658-63, 1996.

[10] Y. Xu, K.W. Au, G.C. Nandy and H.B. Ben, "Analysis of Actuation and Dynamic Balancing for a Single Wheel Robot", *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Vol 4, pp. 3658-63,1998.

[11] G.C. Nandy and Y. Xu, "Dynamic Model of a Gyroscopic Wheel", *Proc. IEEE Int. Conf. on Robotic and Automation*, Vol. 3, pp. 2683-2688, 1998.

[12] Y. Xu and L.W. Sun "Stabilization of a Single Wheel Robot on an Inclined Plane", submitted to *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 1999*

[13] H. Asada and S. Liu, "Transfer of human skills to neural net robot controllers", *Proc. IEEE Int. Conf. on Robotic and Automation*,,1991.

[14] H. Asada and B. Yang, "Skill Acquisition from Human Experts Through Pattern Processing of Teaching Data", *Proc IEEE Int. Conf. on Robotics and Automation*, Vol.3, pp. 1302-1307

[15] S.E. Fahlman, "An Empirical Study of Learning Speed in Back-Propagation Newtworks," *Techanical Report*, CMU-CS-TR-88-162, Carnegie Mellon University, 1988.

[16] S.E. Fahlman and C. Lebiere., "The Cascade-Correlation Learning Algorithm," *Advances in Neural Information Processing System* 2 ed. Touretzky, D.S., Morgan Kaufmann Publishers, PP 524-532, 1990.

[17] S.E. Fahlman, L.D. Baker and J.A. Boyan, "The Cascade 2 learning architecture," *Techanical Report. CMU-CS-TR-96-184, Carnegie Mellon University*,1996.

[18] G.V. Puskorious and L.A. Feldkamp, "Decoupled Extend Kalman Filter Training of Feed Forward Layered Networks", *Proc. Int. Joint Conf. on Neural Networks*, Vol. 1, pp. 771-777, 1991

[19] J.H. Friedman and W. Stuetzle, "Projection Pursuit Regression," *Journal of the American Statistical Association*, Vol. 76, No. 376, pp. 817-823, 1981.

[20] J. Hwang and I. Hang, "A Comparison of Projection Pursuit and Neural Network Regression Modeling," *Advances in Neural Information Processing System* 4 ed. Moody, J.E., Hanson, S.J. and Lippmann, R.P., Morgan Kaufmann Publishers, PP 1159-1166, 1992.

[21] G. Cybenkyo, "Approximation by Superposition of a Sigmoidal Function," *Mathematics of Control, Signals and Systems*, Vol.2, No.4, pp. 303-314, 1989

[22] K. Funahashi, "On the Approximate Realization of Continues Mappings by Neural Networks," *Neural Networks*, Vol. 2, No. 3, pp. 183-192, 1989

[23] K. Hornik, M. Stinchcomb, H. White., "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol. 2, No. 3, pp. 359-366, 1989

[24] Richard L. Gorsuch. *Factor Analysis*, L.Erlbaum Associates, 1983

[25] Alexander Basilevsky. *Statistical Factor Analysis and Related Methods.*, John Wiley and Son, 1994

[26] L.R. Labiner, "A Tutorial On Hidden Markov Models and Selected Applications in Speech Recognition", *Proc. IEEE*, Vol. 77, no. 2, pp 257-86, 1989

[27] L.R. Rabiner, B.H. Juang, S.E. Levinson and M.M. Sondhi, "Some Properties of Continuous Hidden Markov Model Representations," *AT&T Technical Journal*, Vol 64, No. 2, pp. 391-408, 1985

[28] K.S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. of Neural Networks*, Vol 1, no. 1, pp. 4-27, 1990

[29] Y. Linde and A. Buzo and R.M. Gray, "An Algorithm for Vector Quantizer Designer", *IEEE Trans. Communication*, Vol. COM-28, no. 1, pp. 84-95, 1998

[30] W.C. Collier and R.J. Weiland, "Smart Cars, Smart Highways", *IEEE Spectrum*, Vol. 31, No.4, pp 27-33, 1994.

[31] D.Hildebrand, "An architectural overview of QNX", *Proc. Usenix Workshop of Micro-Kernels and Other Kernel Architectures*, Seattle, Aprial, 1992

[32] V.V. Fedorov, *Theory of Optimal Experiments*, Academic Press, New York, 1972.

[33] D.A. White and D.A. Sofge, *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, Multiscience Press, New York, 1992

[34] W.T. Miller, R.S. Sutton and P.I. Werbos, "Neural Networks For Control",MIT Press, Cambridge, 1990

[35] A.G. Barto, R.S. Sutton and C.J. Watkins, *Learning and Sequential Decision Making Learning and Computational Neuroscience*, MIT Press, Cambridge, pp.539-602, 1990

[36] C.C. LEE, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I", *IEEE Trans. System, Man and Cybernetics*, Vol.20, No.2, pp 404-418, 1990.

[37] C.C. LEE, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part II", *IEEE Trans. System, Man and Cybernetics*, Vol.20, No.2, pp 419-435, 1990.

[38] S. Lee adn J. Chen, "Skill Learning from Observation", *Proc IEEE Conf. on Robotics and Automation*, Vol. 4, pp. 3245-3250, 1994

[39] S. S. Ge, *Adaptive neural network control of robotic manipulators* , World Scientific, Singapore, 1998

[40] S. Yasunobu and M.Moris "Swing up Fuzzy Controller for Inverted Pendulum based on Human Control Strategy" *Fuzzy System, Proceedings of the Sixth IEEE International Conference*, Vol. 3, pp 1621-1625, 1997

[41] R.A. Hess, "Human-in-the-loop-control", *The Control Handbook*, ed. W.S. Levine, CRC Press, pp. 1497-1505, 1996

[42] U. Kramer, "On the Application of Fuzzy Sets to the Analysis of the System Driver-Vehicle-Environment", *Automatica*, Vol. 21, No.1, pp. 101-107, 1985

[43] M. Boninsegna and M. Rossi, "Similarity Measure in Computer Vision", *Pattern Recognition Letters*, Vol. 15, No. 12, pp. 1255-1260, 1994

[44] R. Jain, S. N. J. Morty, et. al., "Similarity Measures for Image databases". *Proc. IEEE Int. Conf. on Fuzzy Systems*, Vol. 3, pp. 1247-1254, 1995

[45] K.Y. Kupeev and H. J. Wolfson, "On Shape Similarity" *Proc. of 12th IAPR Int. Conf. on Pattern Recognition*, Vol. 1, pp. 227-231, 1994

[46] B.V. Bonnlander and A.S. Weigend, "Selecting Input Variables Using Mutual Information and Nonparametric Density Estimation", *Proc. of the 1994 International Symposium on Artifical Neural Network (ISANN'94)*, Tainan, Taiwan, 1994, pp42-50.

[47] Andrew D. Back and Thomas p. Trappenberg, "Input Variable Selection Using Independence Component Analysis", *International Joint Conference on Neural Networks (IJCNN)*, Washington DC, July 10-16, 1999