Springer Tracts in Advanced Robotics

Volume 20

Editors: Bruno Siciliano · Oussama Khatib · Frans Groen

Springer Tracts in Advanced Robotics

Edited by B. Siciliano, O. Khatib, and F. Groen

Vol. 19: Lefebvre, T.; Bruyninckx, H.; De Schutter, J.
Nonlinear Kalman Filtering for Force-Controlled Robot Tasks
280 p. 2005 [3-540-28023-5]

Vol. 18: Barbagli, F.; Prattichizzo, D.; Salisbury, K. (Eds.)
Multi-point Interaction with Real and Virtual Objects
281 p. 2005 [3-540-26036-6]

Vol. 17: Erdmann, M.; Hsu, D.; Overmars, M.; van der Stappen, F.A (Eds.)
Algorithmic Foundations of Robotics VI
472 p. 2005 [3-540-25728-4]

Vol. 16: Cuesta, F.; Ollero, A.
Intelligent Mobile Robot Navigation
224 p. 2005 [3-540-23956-1]

Vol. 15: Dario, P.; Chatila R. (Eds.)
Robotics Research – The Eleventh International Symposium
595 p. 2005 [3-540-23214-1]

Vol. 14: Prassler, E.; Lawitzky, G.; Stopp, A.; Grunwald, G.; Hägele, M.; Dillmann, R.; Iossifidis. I. (Eds.)
Advances in Human-Robot Interaction
414 p. 2005 [3-540-23211-7]

Vol. 13: Chung, W.
Nonholonomic Manipulators
115 p. 2004 [3-540-22108-5]

Vol. 12: Iagnemma K.; Dubowsky, S.
Mobile Robots in Rough Terrain – Estimation, Motion Planning, and Control with Application to Planetary Rovers
123 p. 2004 [3-540-21968-4]

Vol. 11: Kim, J.-H.; Kim, D.-H.; Kim, Y.-J.; Seow, K.-T.
Soccer Robotics
353 p. 2004 [3-540-21859-9]

Vol. 10: Siciliano, B.; De Luca, A.; Melchiorri, C.; Casalino, G. (Eds.)
Advances in Control of Articulated and Mobile Robots
259 p. 2004 [3-540-20783-X]

Vol. 9: Yamane, K.
Simulating and Generating Motions of Human Figures
176 p. 2004 [3-540-20317-6]

Vol. 8: Baeten, J.; De Schutter, J.
Integrated Visual Servoing and Force Control
198 p. 2004 [3-540-40475-9]

Vol. 7: Boissonnat, J.-D.; Burdick, J.; Goldberg, K.; Hutchinson, S. (Eds.)
Algorithmic Foundations of Robotics V
577 p. 2004 [3-540-40476-7]

Vol. 6: Jarvis, R.A.; Zelinsky, A. (Eds.)
Robotics Research – The Tenth International Symposium
580 p. 2003 [3-540-00550-1]

Vol. 5: Siciliano, B.; Dario, P. (Eds.)
Experimental Robotics VIII
685 p. 2003 [3-540-00305-3]

Vol. 4: Bicchi, A.; Christensen, H.I.; Prattichizzo, D. (Eds.)
Control Problems in Robotics
296 p. 2003 [3-540-00251-0]

Vol. 3: Natale, C.
Interaction Control of Robot Manipulators – Six-degrees-of-freedom Tasks
120 p. 2003 [3-540-00159-X]

Vol. 2: Antonelli, G.
Underwater Robots – Motion and Force Control of Vehicle-Manipulator Systems
209 p. 2003 [3-540-00054-2]

Vol. 1: Caccavale, F.; Villani, L. (Eds.)
Fault Diagnosis and Fault Tolerance for Mechatronic Systems – Recent Advances
191 p. 2002 [3-540-44159-X]

Yangsheng Xu · Yongsheng Ou

Control of Single Wheel Robots

With 122 Figures and 34 Tables

Springer

Professor Bruno Siciliano, Dipartimento di Informatica e Sistemistica, Università degli Studi di Napoli Federico II, Via Claudio 21, 80125 Napoli, Italy, email: siciliano@unina.it

Professor Oussama Khatib, Robotics Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305-9010, USA, email: khatib@cs.stanford.edu

Professor Frans Groen, Department of Computer Science, Universiteit van Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands, email: groen@science.uva.nl

STAR (Springer Tracts in Advanced Robotics) has been promoted under the auspices of EURON (European Robotics Research Network)

Authors

Yangsheng Xu
Yongsheng Ou

Chinese University of Hong Kong
Department of Automation and Computer-Aided Engineering
Shatin
Hong Kong SAR, P.R. China

ISSN print edition: 1610-7438
ISSN electronic edition: 1610-742X

ISBN-10 3-540-28184-3 **Springer Berlin Heidelberg New York**
ISBN-13 978-3-540-28184-9 **Springer Berlin Heidelberg New York**

Library of Congress Control Number: 2005930322

Springer is a part of Springer Science+Business Media
springeronline.com

Printed in Germany

Typesetting: Digital data supplied by editors.
Data-conversion and production: PTP-Berlin Protago-TEX-Production GmbH, Germany
Cover-Design: design & production GmbH, Heidelberg
Printed on acid-free paper 89/3141/Yu - 5 4 3 2 1 0

To our families

Foreword

At the dawn of the new millennium, robotics is undergoing a major transformation in scope and dimension. From a largely dominant industrial focus, robotics is rapidly expanding into the challenges of unstructured environments. Interacting with, assisting, serving, and exploring with humans, the emerging robots will increasingly touch people and their lives.

The goal of the new series of Springer Tracts in Advanced Robotics (STAR) is to bring, in a timely fashion, the latest advances and developments in robotics on the basis of their significance and quality. It is our hope that the wider dissemination of research developments will stimulate more exchanges and collaborations among the research community and contribute to further advancement of this rapidly growing field.

The monograph written by Yangsheng Xu and Yongsheng Ou is the culmination of a considerable body of research by the first author with the recent support of the second author's Ph.D. dissertation. The work builds upon a novel concept in locomotion of nonholonomic underactuated robots, a field which has lately been attracting more and more scholars. Design, modelling and control of a single-wheel, gyroscopically stabilized robot are explained in detail, and its advantages over multiwheel vehicles are discussed. The volume offers a comprehensive treatment of the subject matter from the theoretical development to experimental testing, while foreseeing a number of potential applications of the new design in service robotics.

Certainly of interest to researchers in mobile robot dynamics and control, this title constitutes a fine addition to the series!

Naples, Italy,
April 2005

Bruno Siciliano
STAR Editor

Preface

The book is a collection of the research work on the control of a single wheel gyroscopically stabilized robot. Although the robot was originally developed at Carnegie Mellon University, most work shown in this book was carried out in the Chinese University of Hong Kong. It focuses on the dynamics and control aspects of the system, including dynamic modeling, model-based control, learning-based control, and shared control with human operators.

The robot is a single wheel balanced by a spinning flywheel attached through a two-link manipulator at a drive motor. The spinning flywheel acts as a gyroscope to stabilize the robot, and at the same time it can be tilted to achieve steering. It shows a completely different picture from conventional mobile robots, opening up the science of dynamically stable, but statically unstable systems.

Conventional robots, either working in industry or service, are statically stable, i.e., the motion is stable when the speed is low, and unstable or malfunctioned when the speed is high. Dynamically stable robots, on the other hand, are getting more and more stable when the speed is increased and tend to be stable even in a rough terrain. The nature of the system is nonholonomic, underactuated, and nonlinear, providing a rich array of research issues in dynamics, control, and sensing, which we have studied in this book to establish a foundation of the science of dynamically stabilized robotics.

Potential applications for it are numerous. Because it can travel on both land and water, it is of amphibious use on beaches or swampy areas, for transportation, rescue, frontier inspections, mining detection, environment monitoring or recreation. As a surveillance robot, it could take advantage of its slim profile to pass through doorways and narrow passages, and its ability of turning in place to maneuver in tight quarters. It also can be used as a high-speed lunar vehicle, where the absence of aerodynamic disturbances and low gravity would permit efficient, high-speed mobility.

The road-map of this book is as follows.

In Chapter 1, a detailed description about the robot is given. We firstly introduce the history of the robot's development. Then, the hardware com-

ponents and the robot's concept are discussed. Finally, a summary of design implementation for the robot is addressed.

In Chapter 2, according to the third prototype of the robot, we derive the kinematics and dynamic model and study its dynamic properties, including the stabilization and the tilt effect of the flywheel to the robot. This chapter is based on the thesis work done by Mr. Samuel Au Kwok-Wai under the supervision of the first author.

In Chapter 3, based on the dynamic model of the robot, we study two classes of nonholonomic constraints associated with the system. We then propose control laws for balance control, point-to-point control and line tracking in Cartesian space. The experimental implementation for verifying the control laws is provided.

Chapter 4 deals with a learning-based approach realized by learning from human expert demonstration, as the model-based control for such a dynamically stable system is too challenging. We then investigate the convergence analysis for this class of learning-based controllers. Last, a method of including new training samples without increasing computational costs is proposed.

Chapter 5 discusses on the input selection topic and the neural network models for the motions of lateral balancing and tiltup implemented experimentally. By combining the two motions into one, the robot is able to recover from the fall position, and then to remain stable at the vertical position after tiltup.

Since autonomous functions in a system may not work perfectly in some unexpected situations, a level of intervention by the human operator is therefore necessary. In Chapter 6, the shared control with human operators, by using the aforementioned autonomous control approaches is investigated. Chapters 5 and 6 are partially an extension of the thesis work done by Mr. Cedric Kwok-Ho Law under the supervision of the first author.

This book is appropriate for postgraduate students, research scientists and engineers with interests in mobile robot dynamics and control. In particular, the book will be a valuable reference for those interested in the topics of mechanical and electrical design and implementation, dynamic modeling for disc-liked mobile robots, and model-based control or learning based control in the context of dynamically stable systems such as unicycle, bicycle, dicycle, motorcycle and legged robots.

We would like to thank Mr. H. Ben Brown, Project Scientist in Robotics Institute at Carnegie Mellon University, USA, for his original contribution to the robot. Mr. Brown, while working with the first author at Carnegie Mellon University, designed and developed several generations of the robot. Although most work shown in this book was carried out in the Chinese University of Hong Kong, it would be impossible without the assistance of Mr. Brown for building the excellent platform for real-time control and various experiments. The first author would also like to take this opportunity to thank Mr. Brown for his long-term support, encouragement and friendship that made his time in Carnegie Mellon more interesting, more exciting, and more meaningful.

Thanks also go to Mr. Samuel Au Kwok-Wai for his preliminary work in the wireless communication and software programming which provides a solid foundation for the control implementation. The authors also extend our thanks to Mr. Huihuan Qian for proofreading the text. We would also like to thank our colleagues for their valuable technical assistance in the final stage of preparing this monograph.

Finally, this book is supported in part by Hong Kong Research Grant Council under the grants CUHK 4403/99E and CUHK 4228/01E.

The Chinese University of Hong Kong,
Summer 2005

Yangsheng Xu and
Yongsheng Ou

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

1.1.1 Brief History of Mobile Robots

Land locomotion can be broadly characterized as quasi-static or dynamic. Quasi-static equilibrium implies that inertial (acceleration-related) effects are small enough to ignore. That is, motions are slow enough that static equilibrium can be assumed. Stability of quasi-static systems depends on keeping the gravity vector through, the center of mass, within the vehicle's polygon of support determined by the ground-contact points of its wheels or feet. Energy input is utilized predominantly in reacting against static forces. Such systems typically have relatively rigid members, and can be controlled on the basis of kinematic considerations.

In dynamic locomotion, inertial forces are significant with respect to gravitational forces. Dynamic effects gain relative importance when speed is high, gravity is weak and dynamic disturbances (e.g. rough terrain) are high. Significant energy input is required in controlling system momentum, and in some cases, in controlling elastic energy storage in the system. As performance limits of mobile robots are pushed, dynamic effects will increasingly come into play. Further, robotic systems that behave dynamically may be able to exploit momentum to enhance mobility, as is clearly demonstrated by numerous human-controlled systems: gymnasts, dancers and other athletes; stunt bicycles and motorcycles; motorcycles on rough terrain; cars that vault obstacles from ramps; etc.

It is paradoxical that those factors which produce static stability may contradict dynamic stability. For example, a four-wheel vehicle that is very low and wide has a broad polygon of support, is very stable statically, and can tolerate large slopes without roll-over. However, when this vehicle passes over bumps, dynamic disturbances at the wheels generate large torques, tending to upset the vehicle about the roll, pitch and yaw axes. In effect, the large

Y. Xu and Y. Ou: Control of Single Wheel Robots, STAR 20, pp. 1–12, 2005.

polygon of support required for static stability provides a leverage mechanism for the generation of dynamic torque disturbances. Further, the support points must comply with the surface, statically as well as dynamically, by control of support points (e.g. suspension) and/or by vehicle attitude changes. Sophisticated vehicle suspensions have been developed to minimize dynamic disturbances, but passive-spring suspensions decrease static stability by allowing the center of mass to move toward the outside of the support polygon. Active suspensions may overcome this problem, but require additional complexity and energy expenditure.

As a second example, consider a bicycle or motorcycle which has two wheels in the fore-aft (tandem) configuration. Such a vehicle is statically unstable in the roll direction, but achieves dynamic stability at moderate speed through appropriate steering geometry and gyroscopic action of the steered front wheel. Steering stability generally increases with speed due to gyroscopic effects. Dynamic forces at the wheel-ground contact point act on or near the vehicle center (sagital) plane, and thus produce minimal roll disturbances. Additionally, the bicycle can remain upright when traveling on side slopes. Thus, sacrificing *static* roll stability enhances the *dynamic* roll stability and permits the vehicle to automatically adjust to side slopes.

As a logical extension of this argument, consider a wheel rolling down an incline. Under the influence of gravity, gyroscopic action causes the wheel to precess (the axis of wheel rotation turns) about the vertical axis–rather than simply falling sideways as it does when not rolling–and the wheel steers in the direction it is leaning. The resulting curved path of motion of the wheel on the ground produces radial (*centrifugal*) forces at the wheel-ground contact point, tending to right the wheel. Dynamic disturbances due to surface irregularities act through or near the wheel's center of mass, producing minimal torques in roll, pitch and yaw. The angular momentum of the wheel, in addition to providing the natural gyroscopic steering mechanism, tends to stabilize the wheel with respect to roll and yaw. In terms of attitude control, the wheel is relatively insensitive to fore/aft and side slopes. The result is a highly stable rolling motion with minimal attitude disturbances and tolerance to fore/aft and vertical disturbances. One can readily observe this behavior by rolling an automobile tire down a bumpy hillside.

There are precedents for single-wheel-like vehicles. In 1869, R.C. Hemmings [43] patented a *Velocipede*, a large wheel encircling the rider, powered by hand cranks. Palmer [82] describes several single-wheel vehicles with an operator riding inside. A 1935 publication [72] describes the *Gyroauto*, which carried the riders between a pair of large, side-by-side wheels, and was claimed capable of a speed of 116 mph (187 kph). Also in [72], there is a description of the *Dyno-Wheel,* a concept having a bus-like chassis straddling a huge central wheel. The relatively large diameter of a single-wheel vehicle enhances its obstacle-crossing ability, smoothness of motion and rolling efficiency [12]. Further, a single-track vehicle can more easily find obstacle-free paths on the ground, and its narrow profile can improve maneuverability. However,

problems with steering, low-speed stability and aerodynamics, have kept such vehicles from becoming commonplace.

1.1.2 Problems of Stable Robots

Traditional Research on mobile robotics has placed heavy emphasis on perception, modeling of the environment and path planning. Consequently, vehicles have been designed to be compatible with these planning limitations, and the need for high speed has not been evident. Ultimately, as sensing and computation capabilities improve, robots will be limited less by planning and more by dynamic factors. Wheeled robots, capable of dynamic behavior - i.e. high-speed motion on rough terrain - and of exploiting vehicle dynamics for mobility, represent an exciting and largely unexplored area.

The purpose of our research is to exploit the natural steering behavior and stability of the rolling wheel in the development of a highly dynamic, single wheel mobile robot. We have built several prototypes of such a vehicle, and demonstrated some of the potential capabilities.

1.1.3 Dynamically Stable Mobile Robots

Researchers have viewed mobile robots largely as quasi-static devices for decades. Numerous robots with four, six or more wheels have been developed to maximize mobility on rough terrain. (See, for example, [13] [30] [45] [52] [55].) Likewise, legged robots, which may have potentially greater mobility, have been built and demonstrated, as described in [60], [110]. Generally these robots have featured low center-of-mass placement and broad base support, along with control and planning schemes designed to keep the center-of-mass gravity vector within the support polygon (e.g. monitoring of slopes, coordination of legs). Many designs have attempted to maximize mobility with large wheels or legs, traction-enhancing tires or feet, multi-wheel driving, large body/ground clearance, articulated body configurations, etc. These robots were often limited by motion-planning constraints and hence designed for low-speed operation, typically 1 kph or less. Dynamic factors have little influence on such systems, and consequently, have been largely ignored.

Traditional research on mobile robotics has placed heavy emphasis on perception, modeling of the environment and path planning. Consequently, vehicles have been designed to be compatible with these planning limitations, and the need for high speed has not been evident. Ultimately, as sensing and computation capabilities improve, robots will be limited less by planning and more by dynamic factors. Wheeled robots, capable of dynamic behavior, i.e. high-speed motion on rough terrain, and of exploiting vehicle dynamics for mobility, represent an exciting and largely unexplored area.

A number of researchers have explored the possibilities of utilizing dynamic behavior in various robot linkages, legged locomotors and other dynamic systems. Examples include Fukuda's *Brachiator* [88] and Spong's robot acrobat

[100], both of which utilized dynamic swinging motions. Koditschek [22] and Atkeson [92] and their students studied the dynamics of juggling. McGeer built a robot that walked down slopes without additional power or control input, utilizing the periodic swinging motion of the legs [70]. Raibert's group built a series of dynamically balanced, hopping and running robots that jumped obstacles, climbed steps [44], and performed flips. Arai [3] and Xu's group [14] developed an underactuated robot system by using the dynamic coupling between the passive and active joints. Dynamic motion in these systems allowed the emergence of behaviors that would not have been possible quasi-statically.

In parallel with the work on linkage dynamics, other researchers have focused on the dynamics and balance of wheeled robots. Vos [109] developed a self-contained unicycle that mimicked the behavior of a human cyclist in maintaining roll and pitch stability. Koshiyama and Yamafuji [59] developed a statically stable, single-wheel robot with an internal mechanism that could could move fore and aft and turn in place; their work emphasized the control of the (non-inverted) pendulum carried on the wheel, utilizing momentum transfer in changing direction.

The invention of the wheel predates recorded history, but many interesting wheeled vehicles have appeared within the pass few centuries. According to the historian [79], the bicycle originated in the late 1700s as an unsteerable vehicle known as the "Hobby Horse". In the early 1800s, steering was added to the front wheel to facilitate changing vehicle direction; serendipitously, the front-steering configuration proved to be self-stabilizing. Prior to this development, the thought of a vehicle traveling stably on two wheels would have been considered ludicrous. In [82], Palmer describes several, single-wheel vehicles. One, an 8-foot diameter, cage-like wheel was driven through foot treadles and hand cranks by the driver inside. Presumably it was stabilized by gyroscopic action, and steered by shifting the driver's weight. A "unicycle", built around 1904, provided a seat for the driver inside a large, annular wheel, powered by a gasoline engine.

The stabilizing effect of gyroscopes has intrigued inventors for centuries, and is central to a number of inventions and patents. Two interesting vehicles appeared in a 1935 publication [47]. The "Gyroauto" carried the driver and a passenger between a pair of large, side-by-side wheels, and was claimed to be capable of attaining a speed of 116 mph. The "Dyno-Wheel" comprised a bus-like chassis straddling a huge central wheel. Outrigger wheels apparently were used to control steering, acceleration and braking. (It is doubtful that the Dyno-Wheel was ever built.) In a 1968 patent [102], Summers proposed gyroscopically stabilizing the roll axis of vehicles, enabling a narrow track for passing through tight spaces (e.g. for logging equipment in forests). Several inventions relate to placing stabilizing gyros on the front [42] or rear [79] wheel of a bicycle. A 1933 patent [111] proposed a gyro stabilizer to reduce shimmy on the front wheels of automobiles.

More recently, Koshiyama, et al and K. Yamafuji, et al [59] developed a single-wheel robot with internal mechanism. The statically stable, spherical

wheel can apparently turn in place through inertial effects of the internal mechanism, giving it good maneuverability. Much of the effort is directed at the control aspects for turning and generating forward motion. D.Vos and A. Flotow [109] at MIT developed an autonomous control scheme for a unicycle that is composed of a wheel, a frame, and turntable (inertia wheel). Two DC motors drive the unicycle wheel for forward/reverse motion, and the turntable for generation of yaw torques. The size and mass of the inertial drive mechanism is large-2 to 3 times that of the wheel itself-limiting maneuverability and the ability to recover from falls. Professor Yuta's group [122] at University of Tsukuba designed a unicycle that has an upper and lower part for steering, and developed algorithms on navigation and control.

Pertinent to the issue of attitude disturbances on wheeled vehicles is the work of Lindemann and Eisen at Jet Propulsion Laboratory [67]. They simulated the dynamic behavior of conventional terrestrial construction equipment, and pointed out the vulnerability of multi-wheeled vehicles to dynamic disturbances. The related research work on multi-wheeled robots can be founded extensively in papers published recently, for example, in [91] and [117].

1.2 Design

1.2.1 Concept and Compromise

Gyrover is a novel, single wheel gyroscopically stabilized robot, originally developed at Carnegie Mellon University [21]. Figure 1 shows a schematic of the mechanism design. Essentially, Gyrover is a sharp-edged wheel, with an actuation mechanism fitted inside the wheel. The actuation mechanism consists of three separate actuators: (1)*a spin motor*, which spins a suspended flywheel at a high rate, imparting dynamic stability to the robot; (2)*a title motor*, which controls the steering of Gyrover; and (3)*a drive motor*, which causes forward and/or backward acceleration, by driving the single wheel directly.

The behavior of Gyrover 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:

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

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. Gyrover supplements this basic concept with the addition of an internal gyroscope — the spinning flywheel

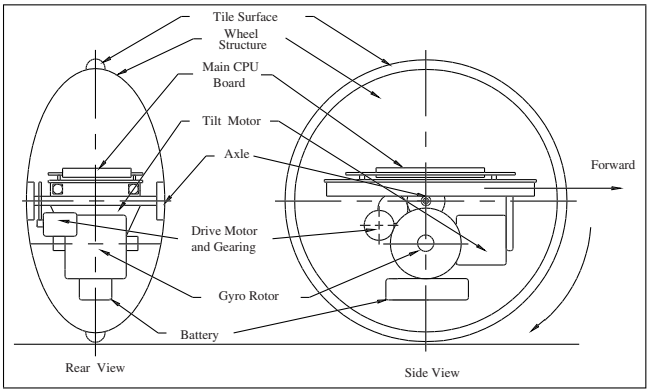


Fig. 1.1. The basic configuration of the robot

— 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.

Gyrover has a number of potential advantages over multi-wheeled vehicles:

1. The entire system can be enclosed within the wheel to provide mechanical and environmental protection for equipment and mechanisms.
2. Gyrover is resistant to getting stuck on obstacles because it has no body to hang up, no exposed appendages, and the entire exposed surface is *live* (driven).
3. The tiltable flywheel can be used to right the vehicle from its statically stable, rest position (on its side). The wheel has no "backside" on which to get stuck.
4. Without special steering mechanism, Gyrover can turn in place by simply leaning and precessing in the desired direction for enhancing maneuverability.
5. Single-point contact with the ground eliminates the need to accommodate uneven surfaces and simplifies control.
6. Full drive traction is available because all the weight is on the single drive wheel.
7. A large pneumatic tire may have very low ground-contact pressure, resulting in minimal disturbance to the surface and minimum rolling resistance. The tire may be suitable for traveling on soft soils, sand, snow or ice; riding over brush or other vegetation; or, with adequate buoyancy, for traveling on water.

Potential applications for Gyrover are numerous. Because it can travel on both land and water, it may find amphibious use on beaches or swampy areas, for general transportation, exploration, rescue or recreation. Similarly, with appropriate tread,it should travel well over soft snow with good traction and

minimal rolling resistance. As a surveillance robot, Gyrover 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. As its development progresses, we anticipate that other, more specific uses will become evident.

1.2.2 Mechanism Design

We have studied the feasibility through basic analysis and simple experiments, and designed and built two, radio-controlled (RC) working models. These have proven the concept workable, and have verified many of the expected advantages.

Given a basic understanding of the gyroscopic principle and wheel dynamic stability, our first task was to find a mechanism for steering the wheel along a desired path. Turning (steering) of the wheel is the result of gyroscopic precession about the yaw axis, caused by roll torques as explained above. We considered two mechanisms for producing this torque: lateral shifting of weight within the vehicle; and leaning of the wheel. With regard to the first approach, the need to provide adequate internal space for shifting large masses within the vehicle is a significant drawback. Moving the mass outside the wheel's envelope is unappealing because of the effective broadening of the vehicle and potential for the movable mass to contact the ground or other obstacles. Allowing the entire wheel to lean employs the complete weight of the wheel to shift laterally – not just a relatively small, movable mass – to generate the needed roll torque. Leaning of the wheel, can be effected by the use of internal reaction wheels or masses, but these tend to acquire kinetic energy, become velocity-saturated, and generate angular momentum that can corrupt vehicle behavior. Another way is to react against an internal gyroscope as described above; this mechanism has been implemented and operated successfully.

Several alternative configurations were considered. A spherical shape, which can be statically stable, does not exhibit the natural steering behavior resulting from the interaction of gravitational (overturning) torque and the gyroscopic effect. In fact, a narrow tire-contact area is desirable for steering responsiveness (dependent on the gravitational torque for a given lean angle). Two wheels, side-by-side provide static stability, but are sensitive to roll disturbances, do not exhibit the same natural steering behavior, and will not roll stably on two wheels above a critical speed. (Observe the behavior of a short cylinder, such as a roll of tape, rolled along the floor.) Outboard wheels, either on the sides or front/back, were considered for static stability and steering effects, as well as acceleration and braking enhancement; but in addition to mechanical complexity, these defeat the basic elegance and control simplicity of the concept. Actually, the robot has the potential to be statically stable to provide a solid base of support for sensors, instruments or small manipulators,

etc., simply by resting on its side. The present concept, totally enclosed in the single wheel, provides a simple, reliable, rugged vehicle.

Experimental work to date includes several simple experiments to verify the stability and steering principle, and three working vehicles.

1.2.3 Sensors and Onboard Computer

The latest model we are using currently is Gyrover III. It is built with a light-weight bicycle tire and rim and a set of transparent domes. It includes a radio system for remote control, on-board computer and a number of sensors to permit data-logging and on-board control of the machine's motion.

There are 3 actuating mechanisms in Gyrover: (i) Gyro tilt servo, (ii) Drive motor, and (iii) DC gyro motor. Table 1.1 gives a detailed description of each of them.

Table 1.1. Table of different actuating mechanisms in Gyrover.

Actuator	Symbol	Descriptions
Gyro tilt servo	u_0	The tilt servo controls the relative angle of the gyro spin axis with respect to the wheel axis. In fact, by controlling the tilt servo, we are able to controls the lean angle angle of the robot indirectly.
Drive motor	u_1	The robot forward/backward drive system uses a 2-stage, tooth belt system to amplify the torque from the drive motor.
DC gyro motor	u_2	This motor cause the internal gyro to spin at a dersirable operating speed, increase the angular momentum of the gyro.

A number of on-board sensors have been installed on Gyrover to provide information about the states of the machine to the on-board computer. The information includes:

- Gyro tilt angle, β_a
- The servo current
- Drive motor current
- Drive motor speed
- Gyro speed, $\dot{\gamma_a}$
- Angular rate (3-axes: Roll-Pitch-Yaw), $\dot{\beta}$, $\dot{\gamma}$ and $\dot{\alpha}$
- Accleration (3-axes: Roll-Pitch-Yaw), $\ddot{\beta}$, $\ddot{\gamma}$ and $\ddot{\alpha}$
- Robot tilt angle (Roll), β

All these signals, plus the control inputs from the radio transmitter, can be read by the computer. A custom-built circuit board contains the control computer and flashdisk, interface circuitry for the radio system and servos,

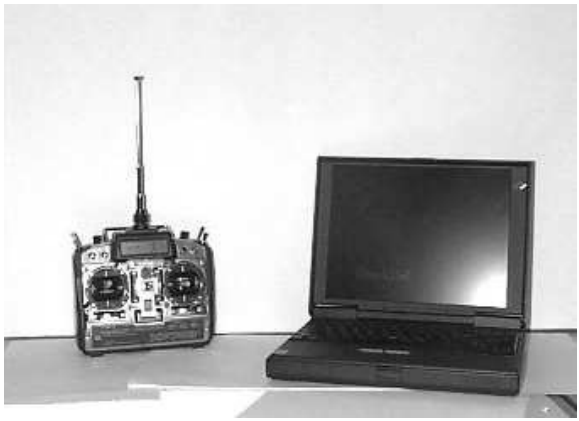


Fig. 1.2. Communication equipment: radio transmitter (left) and laptops with wireless Modem (right).

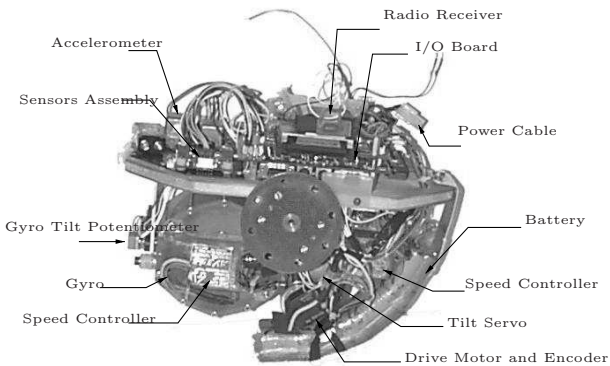


Fig. 1.3. Hardware configuration of the robot.

components and logic to control power for the actuators, and an interface for the on-board sensors. The on-board processing is performed by a 486 Cardio PC.

In addition, several more sensors are planned to be incorporated with our control algorithms in the near future. Visual processing capability or a Global Positioning System (GPS) is a big issue for autonomous control, however, due to the structural limitation of the robot, we have not equipped the robot with this kind of device yet.

An on-board 100-MHZ 486 computer was installed in the robot to deal with on-board sensing and control. A flash PCMCIA card is used as the hard disk of the computer. It communicates with a stationary PC via a pair of wireless modems. Based on this communication system, we can download the sensor data file from the on-board computer, send supervising commands to the robot, and manually control the robot through the stationary PC. More-

over, another radio transmitter is installed for human operators to remotely control the robot via two joysticks of the transmitter(Fig 1.2). One uses the transmitter to control the drive speed and tilt angle of the robot, hence, we can record the operator's driving data.

Numerous sensors are installed in the robot to measure the state variables(Fig 1.3). Two pulse encoders were installed to measure the spinning rate of flywheel and the wheel. Furthermore, we have two gyros and an accelerometer 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 lean angle and pitch angle of the robot. A gyro tilt potentiometer is used to calulate the tilt angle of the flywheel and it's rate change.

The onboard computer is run on an OS, called QNX, which is a real-time microkernel OS developed by QNX Software System Limited. Because of handling numerous sensors and communicating with the stationary PC, the robot's software system is divided into three main programs: (1)communication server, (2)sensor server and (3) controller. The communication server is used to communicate between the onboard computer and the stationary laptop computer via RS232, while a sensor server is used to handle all the sensors and actuators. The controller program implements the control algorithm and communicates among these servers. All these programs are run independently in order to allow real-time control of the robot.

1.2.4 Implementation

The first vehicle, Gyrover I, shown in Figure 1.4, was assembled from RC model airplane/car components, and quickly confirmed the concept. The vehicle has a diameter of 29 cm and mass of 2.0 kg. It can be easily driven and

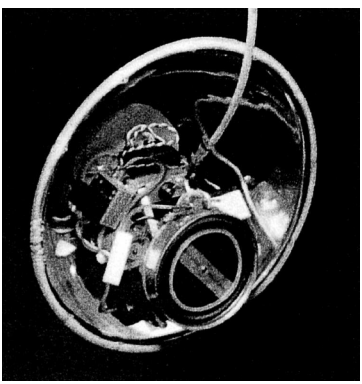


Fig. 1.4. The first prototype of the robot

steered by remote control; has good high-speed stability on smooth or rough terrain; and can be kept standing in place. This vehicle has traveled at over 10 kph, negotiated relatively rough terrain (a small gravel pile), and traversed a 45-degree ramp 75% its height diameter. Recovery from falls (resting on the round side of the wheel) has been achieved with a strategy using both the wheel forward drive and gyro-tilt control. The main shortcomings of this robot are its lack of resilience and vulnerability to wheel damage; excessive battery drain due to drag on the gyro (bearing and aerodynamics); inadequate torque in the tilt servo; and incomplete enclosure of the wheel.

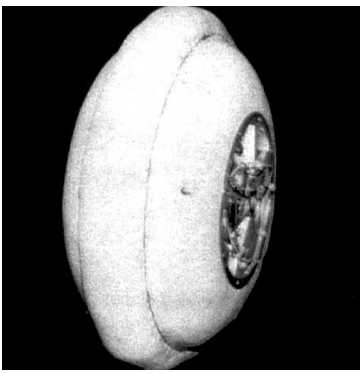


Fig. 1.5. The second prototype of the robot

The second vehicle, Gyrover II, (Figure 1.5) was designed to address these problems. It is slightly larger than Gyrover I (34 cm diameter, 2.0 kg), and also utilizes many RC model parts. Tilt-servo torque and travel were both approximately doubled. Gyrover II uses a gyro housed in a vacuum chamber to cut power consumption by 80%, which increases battery life from about 10 minutes to 50 minutes. The entire robot is housed inside a specially designed pneumatic tire which protects the mechanism from mechanical and environmental abuse, and provides an enclosure that is resilient, although less rugged than hoped. The robot contains a variety of sensors to monitor motor currents, positions and speeds, tire and vacuum pressure, wheel/body orientation, and gyro temperature. Gyrover II has been assembled and driven by manual remote control on a smooth floor, and has shown the ability to float and be controllable on water.

The third version, Gyrover III, (Figure 1.6) was designed on a larger scale to permit it to carry numerous inertial sensors and a computer (486 PC) for data acquisition and/or control. This machine utilizes a lightweight, 40 *cm* bicycle tire and rim, and a pair of transparent domes attached to the axle. Overall weight is about 7 *kg*. Heavy-duty R/C motors and servo are used to spin the gyro, drive the wheel forward and control gyro tilt. Gyrover III trav-

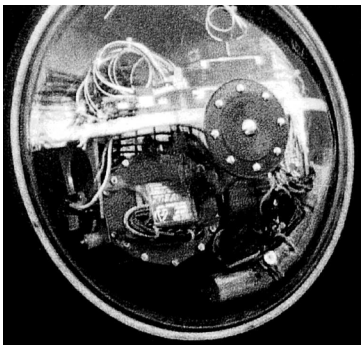


Fig. 1.6. The third prototype of the robot

els up to 10mph, recovers from falls, and runs about 25 minutes per charge of its NiCad batteries. It can carry a video camera that looks through the transparent wheel, and transmit video data to a remote receiver/monitor.

2

Kinematics and Dynamics

2.1 Modeling in a Horizontal Plane

2.1.1 Kinematic Constraints

Previously, Gyrover was controlled only manually, using two joysticks to control the drive and tilt motors through a radio link. A complete dynamic model is necessary to develop automatic control of the system. In the following sections, we will develop the nonholonomic kinematics constraints, as well as a dynamic model using the constrained generalized Lagrangian formulation.

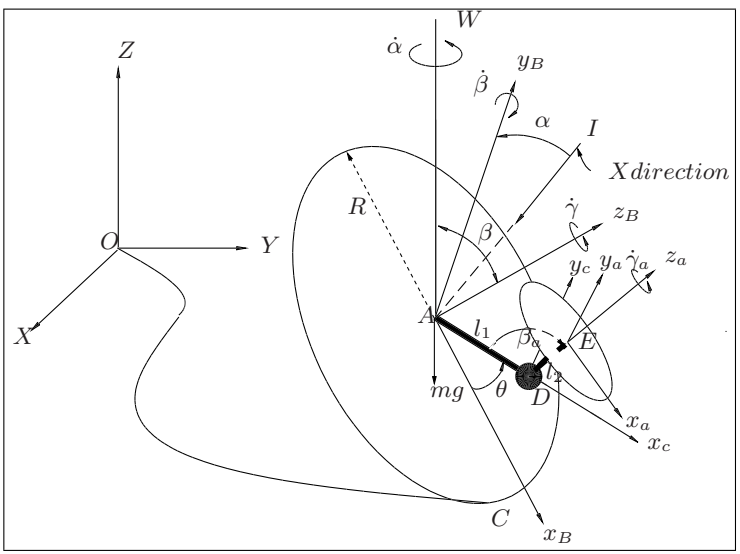


Fig. 2.1. Definition of coordinate frames and system variables

Y. Xu and Y. Ou: Control of Single Wheel Robots, STAR 20, pp. 13–32, 2005.

Coordinate frame

In deriving the equations of motion of the robot, we assume that the wheel is a rigid, homogeneous disk which rolls over a perfectly flat surface without slipping. We model 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). 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.

Table 2.1. Variables definitions

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_1, u_2	Drive torque of the drive motor and tilt torque of the tilt motor, respectively

Next, we assign four coordinates frames 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 2.1 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 nonholonomic system. Here we first

derive the constraints of the single wheel, and then derive the dynamic model of Gyrover based on these constraints. We define (i, j, k) and (l, m, n) 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

$$\begin{bmatrix} i \\ j \\ k \end{bmatrix} = R_B^O \begin{bmatrix} l \\ m \\ n \end{bmatrix} \tag{2.1}$$

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

$$R_B^O = \begin{bmatrix} -S_\alpha C_\beta & -C_\alpha & -S_\alpha S_\beta \\ C_\alpha C_\beta & -S_\alpha & C_\alpha S_\beta \\ -S_\beta & 0 & C_\beta \end{bmatrix} \tag{2.2}$$

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

$$\omega_B = -\dot{\alpha} S_\beta l + \dot{\beta} m + (\dot{\gamma} + \dot{\alpha} C_\beta) n \tag{2.3}$$

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

$$v_c = 0, \tag{2.4}$$

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

$$v_A = \omega_B \times r_{AC} + v_c \tag{2.5}$$

where $r_{AC} = -Rl$ representing the vector from the frame C to A in Figure 5. Substituting Eqs. (2.3) and (2.4) in Eq. (2.5) gives

$$v_A = \dot{X} i + \dot{Y} j + \dot{Z} k, \tag{2.6}$$

where

$$\dot{X} = R(\dot{\gamma} C_\alpha + \dot{\alpha} C_\alpha C_\beta - \dot{\beta} S_\alpha S_\beta) \tag{2.7}$$

$$\dot{Y} = R(\dot{\gamma} S_\alpha + \dot{\alpha} C_\beta S_\alpha + \dot{\beta} C_\alpha S_\beta) \tag{2.8}$$

$$\dot{Z} = R\dot{\beta} C_\beta \tag{2.9}$$

Eqs. (2.7) and (2.8) are nonintegrable and hence are nonholonomic while Eq. (2.9) is integrable, i.e,

$$Z = RS_\beta. \tag{2.10}$$

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

2.1.2 Equations of Motion

In this section, we study the equation of motion 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. We divide the system into three parts: 1) single wheel, 2) internal mechanism, and 3) spinning flywheel.

Single wheel

The kinetic energy of the single wheel is given by,

$$\begin{aligned} T_w &= \frac{1}{2} m_w \left[\dot{X}^2 + \dot{Y}^2 + \dot{Z}^2\right] \\ &\quad + \frac{1}{2}\left[I_{xxw}\omega_x^2 + I_{yyw}\omega_y^2 + I_{zzw}\omega_z^2\right] \end{aligned} \tag{2.11}$$

Substituting Eqs.(2.3) and (2.9) in Eq.(2.11) yields

$$\begin{aligned} T_w &= \frac{1}{2} m_w \left[\dot{X}^2 + \dot{Y}^2 + (R\dot{\beta}C_\beta)^2\right] + \frac{1}{2}\left[I_{xxw}(\dot{\alpha}S_\beta)^2 \right. \\ &\quad \left. + I_{yyw}\dot{\beta}^2 + I_{zzw}(\dot{\alpha}C_\beta + \dot{\gamma})^2\right] \end{aligned} \tag{2.12}$$

The potential energy of the single wheel is

$$P_w = m_w g R S_\beta \tag{2.13}$$

Internal mechanism and spinning flywheel

We need to compute the translational and rotational parts of kinetic energy for the internal mechanism and flywheel respectively. We assume that l_2 is very small compared with l_1,

$$l_2 \simeq 0 \tag{2.14}$$

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 by:

$$\begin{bmatrix} x_f \\ y_f \\ z_f \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + R_B^O \begin{bmatrix} l_1 C_\theta \\ l_1 S_\theta \\ 0 \end{bmatrix} \tag{2.15}$$

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

$$T_f^t = \frac{1}{2}(m_i + m_f)[\dot{x}_f^2 + \dot{y}_f^2 + \dot{z}_f^2] \tag{2.16}$$

Differentiating Eq. (2.15) and substituting it in Eq. (2.16), we obtain T_f^t. We observed that the internal mechanism swings slowly, so it should not contribute highly to the rotational kinetic energy. Let ω_f be the angular velocity of flywheel w.r.t. $\sum_O$. We then have

$$\omega_f = R_B^E \omega_B + \begin{bmatrix} 0 \\ \dot{\beta}_a \\ \dot{\gamma}_a \end{bmatrix} \tag{2.17}$$

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

$$R_B^E = \begin{bmatrix} C_\theta S_{\beta_a} & -S_\theta S_{\beta_a} & C_{\beta_a} \\ S_\theta & C_\theta & 0 \\ C_\theta C_{\beta_a} & -C_{\beta a} S_\theta & S_{\beta_a} \end{bmatrix} \tag{2.18}$$

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

$$T_f^r = \frac{1}{2}\left[(\omega_{fx})^2 I_{xxf} + (\omega_{fy})^2 I_{yyf} + (\omega_{fz})^2 I_{zzf}\right] \tag{2.19}$$

The flywheel is assumed to be a uniform disk and 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

$$P_f = (m_i + m_f)(RS_\beta - l_1 C_\theta S_\beta) \tag{2.20}$$

Lagrangian of the system

The Lagrangian of the system thus is

$$L = \left[T_w + (T_f^t + T_f^r)\right] - (P_w + P_f) \tag{2.21}$$

Substituting Eqs. (2.11), (2.16), (2.19), (2.13) and (2.20) in Eq. (2.21), 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

$$M(q)\ddot{q} + N(q, \dot{q}) = A^T \lambda + Bu \tag{2.22}$$

where $M(q) \in R^{7\times 7}$ and $N(q, \dot{q}) \in R^{7\times 1}$ are the inertia matrix and nonlinear terms respectively.

$$A(q) = \begin{bmatrix} 1 & 0 & -RC_\alpha C_\beta & RS_\alpha S_\beta & -RC_\alpha & 0 & 0 \\ 0 & 1 & -RC_\beta S_\alpha & -RC_\alpha S_\beta & -RS_\alpha & 0 & 0 \end{bmatrix} \tag{2.23}$$

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

The nonholonomic constraints can be written as,

$$A(\dot{q})\dot{q} = 0. \tag{2.24}$$

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, *matrix B only has three rows with non-zero 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 matrix B is not zero.

Normal form of the system

In this section, we will eliminate the Lagrange multipliers so that a minimum set of differential equations is obtained by the similar way in [18]. We first partition the matrix $A(q)$ into A_1 and A_2 where $A = [A_1 : A_2]$

$$A_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, A_2 = \begin{bmatrix} -RC_\alpha C_\beta & RS_\alpha S_\beta & -RC_\alpha & 0 & 0 \\ -RC_\beta S_\alpha & -RC_\alpha S_\beta & -RS_\alpha & 0 & 0 \end{bmatrix}$$

Let

$$C(q) = \begin{bmatrix} -A_1^{-1}A_2 \\ I_{3\times 3} \end{bmatrix} \tag{2.25}$$

Then consider the following relationship

$$\dot{q} = C(q)\dot{q}_2 \tag{2.26}$$

where $q_1 = [X, Y]^T$ and $q_2 = [\alpha, \beta, \gamma, \beta_a, \theta]^T$. Differentiating Eq. (2.26) yields

$$\ddot{q} = C(q)\ddot{q}_2 + \dot{C}(q)\dot{q}_2 \tag{2.27}$$

Substituting Eq. (2.27) into Eq. (2.37) and premultiplying both sides by $C^T(q)$ gives

$$\begin{aligned} &C^T(q)M(q)C(q)\ddot{q}_2 \\ &= C^T(q)\left[Bu - N(q, C(q)\dot{q}_2) - M(q)\dot{C}(q)\dot{q}_2\right] \end{aligned} \tag{2.28}$$

where $C^T(q)M(q)C(q)$ is 5×5 a symmetrical positive definite matrix function. Note that *Eq.(2.28) depends only on* $(\alpha, \beta, \gamma, \beta_a, \theta)$. By numerical integration, we can obtain q_2 from $\ddot{q}_2$ in Eq. (2.28), and then obtain (X, Y) by substituting q_2 and $\dot{q}_2$ in Eq. (2.26).

2.1.3 Dynamic Properties

Understanding the characteristics of the robot dynamics is of significance in the control of the system. To this end, we first simplify the model. Practically, we may assume $I_{xxw} = I_{yyw} = \frac{1}{2}m_wR^2$, $I_{zzw} = m_wR^2$, and l_1 and l_2 are zero, thus the mass center of the flywheel is coincident with the center of the robot. For steady motion of the robot, the pendulum motion of the internal mechanism is sufficiently small to be ignored, thus θ is set to be zero. The spinning rate of the flywheel γ_a is set to be constant. Let $S_{\beta,\beta_a} := sin(\beta + \beta_a)$, $C_{\beta,\beta_a} := cos(\beta+\beta_a)$, and $S_{2\beta\beta_a} := sin[2(\beta+\beta_a)]$. Based on the previous derivation, the *normal form* of the dynamics model is

$$M(q)\ddot{q} = F(q,\dot{q}) + Bu \tag{2.29}$$

$$\dot{X} = R(\dot{\gamma}C_\alpha + \dot{\alpha}C_\alpha C_\beta - \dot{\beta}S_\alpha S_\beta) \tag{2.30}$$

$$\dot{Y} = R(\dot{\gamma}S_\alpha + \dot{\alpha}C_\beta S_\alpha + \dot{\beta}C_\alpha S_\beta) \tag{2.31}$$

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

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

$$F = [F_1, F_2, F_3, F_4]^T,$$

$$B = \begin{bmatrix} 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 1 \end{bmatrix}^T, u = \begin{bmatrix} u1 \\ u2 \end{bmatrix}$$

$$\begin{aligned}
M_{11} &= I_{xxf} + I_{xxw} + I_{xxw}C_\beta^2 + mR^2C_\beta^2 + I_{xxf}C_{\beta,\beta_a}^2 \\
M_{13} &= 2I_{xxw}C_\beta + mR^2C_\beta \\
F_1 &= (I_{xxw} + mR^2)S_{2\beta}\dot{\alpha}\dot{\beta} + I_{xxf}S_{2\beta\beta_a}\dot{\alpha}\dot{\beta} + I_{xxf}S_{2\beta\beta_a}\dot{\alpha}\dot{\beta}_a \\
&\quad +2I_{xxw}S_\beta\dot{\beta}\dot{\gamma} + 2I_{xxf}S_{\beta,\beta_a}\dot{\beta}\dot{\gamma}_a + 2I_{xf}S_{\beta,\beta_a}\dot{\beta}_a\dot{\gamma}_a - \mu_s\dot{\alpha} \\
F_2 &= -gmRC_\beta - (I_{xxw} + mR^2)C_\beta S_\beta\dot{\alpha}^2 - I_{xxf}C_{\beta,\beta_a}S_{\beta,\beta_a}\dot{\alpha}^2 \\
&\quad -(2I_{xxw} + mR^2)S_\beta\dot{\alpha}\dot{\gamma} - 2I_{xxf}S_{\beta,\beta_a}\dot{\alpha}\dot{\gamma}_a \\
F_3 &= 2(I_{xxw} + mR^2)S_\beta\dot{\alpha}\dot{\beta} \\
F_4 &= -I_{xxf}C_{\beta,\beta_a}S_{\beta,\beta_a}\dot{\alpha}^2 - 2I_{xxf}S_{\beta,\beta_a}\dot{\alpha}\dot{\gamma}_a
\end{aligned}$$

where (X, Y, Z) is the coordinates of the center of mass of the single wheel with respect to the inertial frame as shown in Figure 2.1. $M(q) \in R^{4\times4}$ and $N(q,\dot{q}) \in R^{4\times1}$ are the inertial matrix and nonlinear term respectively. Eqs.

(2.29) and (2.30), (2.31) form the dynamics model and nonholonomic velocity constraints of the robot.

We further simplify the model by decoupling the tilting variable β_a from Eq. (2.29). Practically, β_a is directly controlled by the tilt motor (position control), assuming that the tilt actuator has an adequate torque to track the desired $\beta_a(t)$ trajectory exactly. Therefore, β_a can be decoupled from Eq. (2.29). It is similar to the case of decoupling the steering variable from the bicycle dynamics shown in [16],[35].

As we consider $\dot{\beta}_a$ as a new input u_{β_a}, the dynamics model Eq. (2.29) becomes

$$\begin{aligned} \dot{\beta}_a &= u_{\beta_a} \\ \tilde{M}(\tilde{q})\ddot{\tilde{q}} &= \tilde{F}(\tilde{q}, \dot{\tilde{q}}) + \tilde{B}\tilde{u}. \end{aligned} \tag{2.32}$$

with $\tilde{q} = [\alpha, \beta, \gamma]^T$,

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

$$\tilde{F} = \left[\tilde{F}_1, \tilde{F}_2, \tilde{F}_3\right]^T$$

$$\tilde{B} = \begin{bmatrix} 0 & 0 & 1 \\ \tilde{B}_{12} & 0 & 0 \end{bmatrix}^T, \tilde{u} = \begin{bmatrix} u_1 \\ u_{\beta_a} \end{bmatrix}$$

$$\begin{aligned} \tilde{F}_1 &= (I_{xxw} + mR^2) S_{2\beta} \dot{\alpha}\dot{\beta} + I_{xxf} S_{2\beta\beta_a} \dot{\alpha}\dot{\beta} \\ &\quad -2I_{xxw} S_{\beta} \dot{\beta}\dot{\gamma} + 2I_{xxf} S_{\beta,\beta_a} \dot{\beta}\dot{\gamma}_a - \mu_s \dot{\alpha}, \\ \tilde{F}_2 &= F_2, \\ \tilde{F}_3 &= F_3, \\ \tilde{B}_{12} &= I_{xxf} S_{2\beta\beta_a} \dot{\alpha} + 2I_{xxf} S_{\beta,\beta_a} \dot{\gamma}_a, \end{aligned}$$

Eq. (2.32) shows the reduced dynamic model of the single wheel robot after decoupling the tilting varible β_a, with new matrices $\tilde{M}(\tilde{q}) \in R^{3\times3}$ and $\tilde{F}(\tilde{q}, \dot{\tilde{q}}) \in R^{3\times1}$. The realistic geometric/mass parameters are shown in Table 2.2. It is noted that if the lean angle β is set to be 0^o or 180^o, meaning that the single wheel robot falls on the ground, its inertial matrix $\tilde{M}$ will become singular because it violates the assumption of rolling without slipping.

The robot dynamic model is highly coupled, nonholonomic and underactuacted. Because the major part of the robot is a rolling wheel, therefore it processes the typical characteristics of a rolling disk. For a rolling disk, it does not fall when it is rolling, because there is a gyroscopic torque, resulting from the coupling motion between the roll and yaw motions, for balancing the gravitational torque. However, its rolling rate must be high enough to provide a sufficient gyroscopic torque for balancing the disk. When we installed

Table 2.2. Parameters used in simulation and experiments

Single wheel parameters:	$m = 1.25kg,\ R = 17cm$
Internal mechanism :	$m_i = 4.4kg,$
Flywheel parameters :	$m_f = 2.4kg,\ r = 5cm$
Friction coefficients :	$\mu_s = 1Nm/(rad/s),\ \mu_g = 0.01Nm/(rad/s)$

a flywheel on the wheel, the robot's gyroscopic torque is greater and depends less on its rolling speed $\dot{\gamma}$, owing to the high spinning flywheel, the lean angle of the robot tends to remain unchanged. We can explain this characteristics based on the equilibrium solution of Eq. (2.32). For the low yaw rate and $\ddot{\alpha} = \ddot{\beta} = \ddot{\gamma} = 0, \dot{\beta} = 0, u_1 = u_{\beta_a} = 0$, Eq. (2.32) becomes

$$0 = gmRC_\beta + (2I_{xxw} + mR^2)S_\beta \dot{\alpha}\dot{\gamma} + 2I_{xxf}S_{\beta,\beta_a}\dot{\alpha}\dot{\gamma}_a, \tag{2.33}$$

From Eq. (2.33), the terms $\dot{\gamma}\dot{\alpha}$ and $\dot{\gamma}_a\dot{\alpha}$ are used to cancel the gravitational torques $gmRC_\beta$, in order to stabilize the roll component of the system. Because the spinning rate $\dot{\gamma}_a$ is very high, the term $2I_{xxf}S_{\beta,\beta_a}\dot{\alpha}\dot{\gamma}_a$ is significantly large, in order to cancel the gravitational torque, even though the rolling speed $\dot{\gamma}$ is low. Thus, it will achieve an equilibrium steering rate $\dot{\alpha}_s$,

$$\dot{\alpha}_s = \frac{-gmRC_{\beta_s}}{2I_{xxf}S_{\beta_s,\beta_a}\dot{\gamma}_a + (2I_{xxw} + mR^2)S_{\beta_s}\dot{\gamma}} \tag{2.34}$$

for a specific lean angle β_s.

Figures 2.2 and 2.3 above show the simulation results of a rolling disk without the flywheel, and that of the single wheel robot, respectively, under the same initial conditions

$$\begin{cases} \beta = 70^o, \beta_a = 0^o, \\ \dot{\beta} = \dot{\alpha} = \dot{\beta}_a = 0\ rad/s, \dot{\gamma} = 15\ rad/s, \\ \alpha = 0^o. \end{cases}$$

Note that the lean angle β of a rolling disk without the flywheel decreases much rapidly than that of the single wheel robot as shown in Figures 2.2(b) and 2.3(b). This verifies the stabilizing effect of the flywheel on the single wheel robot. In Figures 2.3(a),(c), under the influence of friction in the yaw direction, the steering rate $\dot{\alpha}$ and the leaning rate $\dot{\beta}$ will converge to a steady state solution as shown in Eq. (2.34). Otherwise, an unwanted high frequency oscillation will occur. Practically, it will not produce a significant effect on the system because the frequency of the above oscillation is much greater than the system response. If the rolling rate is reduced, the rolling disk will fall much more quickly than in the previous case.

2.1.4 Simulation Study

Up to now, we only consider the case when the flywheel's orientation is fixed with respect to the single wheel. Here, we will focus on the tilting effect of the

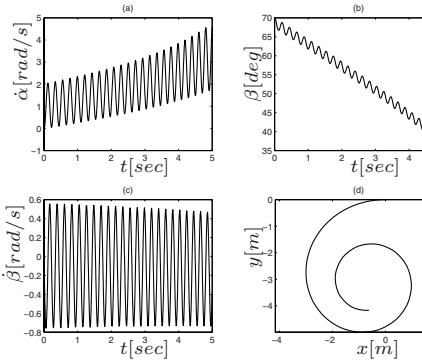


Fig. 2.2. The simulation results of a rolling disk without the flywheel.

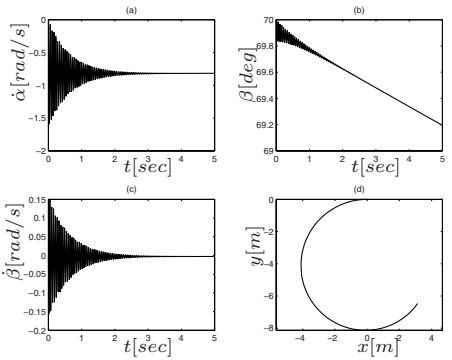


Fig. 2.3. The simulation results of the single wheel robot.

flywheel on the robot. Based on the conservation of angular momemtum, when there is a change in the tilt angle of the flywheel β_a, the whole robot will rotate in the *opposite direction* in order to maintain constant angular momentum. It implies that we may control the lean angle of the robot for steering. Simulation and experiment results are shown in Figure 2.4 and Figure 2.5, respectively, under the same initial conditions

$$\begin{cases} \beta = 90^o, \beta_a = \alpha = 0^o, \\ \dot{\beta} = \dot{\alpha} = \dot{\beta}_a = 0 \ rad/s, \dot{\gamma} = 15 \ rad/s, \\ \alpha = 0^o. \end{cases}$$

Both Figures 2.4 and 2.5 show that if the tilt angle β_a rotates in 73 deg/s anticlockwise at t = 2.4 seconds, the lean angle β rotates in a clockwise direction. In the experiment, the transient response of β is more critical than the simulations. For 2.7 seconds, the tilt angle remains unchanged and then β and steering rate $\dot{\alpha}$ converge to a steady position in both simulations and experiments. In the experiment, we have not found any high frequency oscillations perhaps because the sensor response and the sampling time are much slower than the high frequency oscillations.

2.2 Modeling on an Incline

2.2.1 Motion on an Incline

Dynamics of a rolling disk

Consider a disk rolls without slipping along an inclined plane with an inclination angle φ. Assume that the disk is rigid with radius R. The dynamic model can be developed using the constrained Lagrangian method. It is similar to

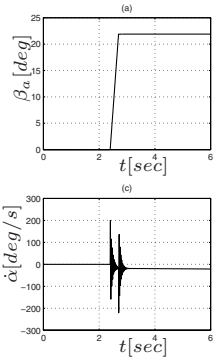


Fig. 2.4. The simulation results of tilting the flywheel of the robot with $\dot{\beta}_a = 73\ deg/s$

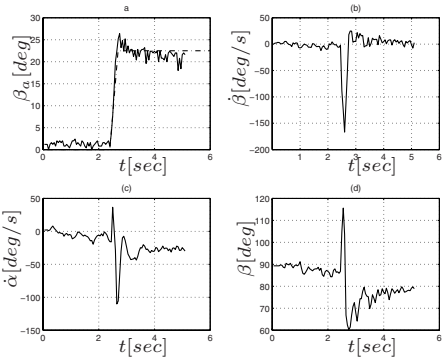


Fig. 2.5. The experimental results of tilting the flywheel of the robot with $\dot{\beta}_a = 73\ deg/s$

the derivation in [115] & [87], except for some components due to the gravity act on the system. Two nonholonomic velocity constraints are

$$\dot{X} = -R\big(\dot{\gamma}S_\alpha + \dot{\alpha}S_\beta C_\alpha - \dot{\beta}C_\alpha S_\beta\big) \tag{2.35}$$

$$\dot{Y} = -R\big(\dot{\gamma}C_\alpha + \dot{\alpha}C_\beta C_\alpha + \dot{\beta}S_\alpha S_\beta\big) \tag{2.36}$$

The rotational kinetic energy and the potential energy of the disk are given below,

$$\begin{aligned}\mathcal{K} &= \frac{1}{2}\Big[m\dot{X}^2 + m\dot{Y}^2 + mR^2\dot{\beta}^2\cos^2\beta\Big] + \frac{1}{2}I_x\dot{\alpha}^2\sin^2\beta \\ &\quad + \frac{1}{2}I_x\dot{\beta}^2 + I_x\big(\dot{\alpha}\cos\beta + \dot{\gamma}^2\big) \\ \mathcal{P} &= mg(R\sin\beta\cos\varphi + y\sin\varphi)\end{aligned}$$

Using the Lagrange undetermined multiplier λ method, the equations of motion of a rolling disk can be determined below

$$\begin{aligned} m\ddot{X} &= \lambda_1 \\ m\ddot{Y} + mg\sin\varphi &= \lambda_2 \\ \lambda_1 Rs\alpha + \lambda_2 Rc\alpha &= 2I_x\big(\ddot{\alpha}c\beta + \ddot{\gamma} - \dot{\alpha}\dot{\gamma}s\beta\big) \\ \lambda_1 Rs\alpha c\beta + \lambda_2 rc\alpha c\beta &= I_x\big(\ddot{\alpha}(1+c^2\alpha) - 2\dot{\alpha}\dot{\beta}s\beta c\beta \\ &\quad + 2\ddot{\gamma}c\beta - 2\dot{\beta}\dot{\gamma}s\beta\big) \\ C - \lambda_1 Rc\alpha s\beta + \lambda_2 Rs\alpha s\beta &= \big(mr^2c^2\alpha + I_x\big)\ddot{\beta} - mR^2\dot{\beta}^2c\beta \\ &\quad + mgRc\beta + I\dot{\alpha}^2c\beta s\beta + 2I_x\dot{\alpha}\dot{\gamma}s\beta \end{aligned}$$

where $c = \cos$ and $s = \sin$. We eliminate the Lagrange multipliers by obtaining $\ddot{X}, \ddot{Y}$. A normal form of the dynamic equation of a rolling disk is given by

$$M(q)\ddot{q} + N(q, \dot{q}) = 0$$

where $M(q) \in \mathbb{R}^{3\times3}$ and $N(q, \dot{q}) \in \mathbb{R}^{3\times1}$ are the inertia matrix and nonlinear terms respectively.

$$M = \begin{bmatrix} I_x + (mR^2 + I_x)c^2\beta & 0 & (mR^2 + 2I_x)c\beta \\ 0 & mR^2 + I_x & 0 \\ (mR^2 + 2I_x)c\beta & 0 & (mR^2 + 2I_x) \end{bmatrix}$$

$$\begin{aligned} N &= [N_1, N_2, N_3]^T, \quad q = [\alpha, \beta, \gamma]^T \\ N_1 &= -mgRc\alpha c\beta s\varphi - (I_x + mR^2)s(2\beta)\dot{\alpha}\dot{\beta} - 2I_x s\beta\dot{\beta}\dot{\gamma} \\ N_2 &= mgRc\beta c\varphi - gmRs\alpha s\beta s\varphi + (mr^2 + I_x)c\beta s\beta\dot{\alpha}^2 \\ &\quad +(mR^2 + I_x)s\beta\dot{\alpha}\dot{\gamma} \\ N_3 &= -2(mR^2 + I_x)s\beta\dot{\alpha}\dot{\beta} \end{aligned}$$

Dynamics of a single wheel robot

The motion of a single wheel robot on an inclined plane differs significantly from that on horizontal planes [21],[115]&[112]. On an incline, some components of the gravitational forces act on the system. The nature of nonholonomic constraints and the equation of motion can be derived as below. We assume that the robot is a homogeneous, rigid disk which rolls over a sufficiently rough slope without slipping. The high spinning flywheel is anchored on the centre of mass of the entire wheel. The dynamic equation of the entire system is given by

$$M_6(q)\ddot{q} + N_6(q, \dot{q}) = A^T\lambda + Bu \tag{2.37}$$

where $M_6(q) \in \mathbb{R}^{6\times6}$ and $N_6(q, \dot{q}) \in \mathbb{R}^{6\times1}$ are the inertia matrix and nonlinear terms respectively.

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

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T, u = [u_1, u_2]^T$$

$$q = [X, Y, \alpha, \beta, \gamma, \beta_a]^T, \lambda = [\lambda_1, \lambda_2]^T$$

The nonholonomic constraints of a rolling disk and a single wheel robot on the inclined plane are identifical in Equations 2.35 & 2.36. The nonholonomic constraints can be written as,

$$A(q)\dot{q} = 0 \tag{2.38}$$

A minimum set of differential equations (Normal form) is obtained when the Lagrange multipliers are eliminated. This model (2.39) is a nonholonomic and underactuated model. The model of a single wheel robot on the ground [112], assumed that the climbing angle $\varphi = 0$, is a subset of this model. The system dynamics can be described as

$$\begin{bmatrix} m_{11} & 0 & m_{13} & 0 \\ 0 & m_{22} & 0 & m_{24} \\ m_{13} & 0 & m_{33} & 0 \\ 0 & m_{24} & 0 & m_{44} \end{bmatrix} \begin{bmatrix} \ddot{\alpha} \\ \ddot{\beta} \\ \ddot{\gamma} \\ \ddot{\beta}_a \end{bmatrix} + N(q, \dot{q}) \;=\; Bu \tag{2.39}$$

2.2.2 Motion Planning on an Incline

Condition of rolling up

In this section, we determine the condition of rolling up of the robot on an inclined plane. The system can be linearized around the position perpendicular to the surface such as $\beta = 90^\circ + \delta\beta, \beta_a = \delta\beta_a, \dot{\beta} = \dot{\delta\beta}$. After linearization, the linear acceleration of the system along the plane and the angular acceleration of the rolling disk are

$$\ddot{\gamma} = \frac{u_1 - mgRC_\alpha S_\varphi}{mr^2 + I}$$
$$\ddot{Y} = \frac{u_1 - mgRC_\alpha S_\varphi}{mr^2 + I} R$$

where I represents the moment of the whole robot along the Z axis, $2I_{xw}$. Consider a rolling disk, I is set to be the moment of inertia of the disk along the Z axis. The condition of rolling without slipping holds, i.e., $\ddot{y} = R\ddot{\gamma}$.

Initially $\dot{\gamma}, v_0$ are set to be zero. The minimum value of the angular acceleration of the system is set to be $\ddot{\gamma}_{min}$. Therefore, the condition of rolling up on an incline is

$$u_1 \geq (mR^2 + I)\ddot{\gamma}_{min} + mgR\sin\varphi$$

Let's rearrange the above equation so that φ is represent as a function of the minimum angular acceleration of the system $\ddot{\gamma}_{min}$, the moment of inertia I and the applied torque u_1.

$$\sin\varphi = \frac{u_1 - (mR^2 + I)\ddot{\gamma}_{min}}{mgR} \tag{2.40}$$

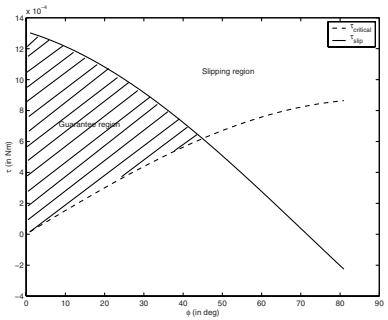


Fig. 2.6. Critical torque of a rolling disk v.s. a climbling angle

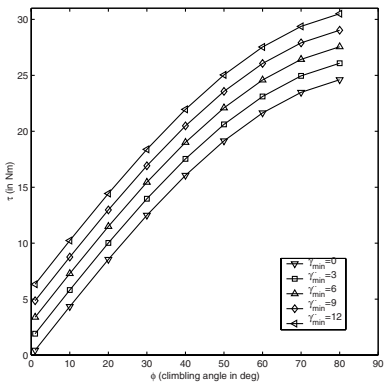


Fig. 2.7. Critical torque of the robot v.s. a climbling angle

Figures. 2.6 & 2.7 show the critical torques applied to the system for rolling up an inclined plane with an inclination φ. Referring to Fig. 2.7, the condition of rolling up for a single wheel robot is similar to a rolling disk in Fig.2.6. The curve of a critical torque looks similar to that of a rolling disk, but it has larger magnitude.

The dynamic balancing

We have proposed a linear state backstepping feedback[113]. Consider $x_1 = \delta\beta, x_2 = \dot{\alpha}, x_3 = \dot{\delta\beta}$, the systems can be described in a different form:

$$\begin{aligned} \dot{x_1} &= x_3 \\ \dot{x_2} &= f_{21}x_1 + f_{22}x_2 + f_{23}x_3 + g_{21}u_{\beta_a} \\ \dot{x_3} &= f_{31}x_1 + f_{32}x_2 + g_{32} \end{aligned} \tag{2.41}$$

We define the following control law,

$$u_{\beta_a} = \frac{-f_{21}x_1 - f_{22}x_2 - f_{23}x_3 + \dot{\alpha_2} - k_3(x_2 - \frac{\eta}{f_{32}})}{g_{21}}$$
$$\dot{\alpha_2} = \frac{-(1+f_{31})\dot{x_1} - k_1\dot{x_3} - k_2(x_3 + k_1x_1)}{f_{32}}$$

which can ensure the system to be stabilized around the position perpendicular to the surface.

Planning rolling Up

In this section, we discuss how to remedy the failure of the condition of rolling up by planning the robot motion in different angles.

Method I : Changing of the orientation. Suppose we change the direction of heading

$$\sin\phi = \frac{y\sin\varphi}{\sqrt{x^2+y^2}} = \sin\varphi\sin\alpha$$

From the previous result, the condition of rolling up is

$$C \geq (mR^2 + I)\ddot{\gamma}_{min} + mgR\sin\alpha\sin\varphi$$

Now we consider the robot rolls on an inclined plane ϕ instead of φ.

$$\frac{1}{2}I\ddot{\alpha} = 0 \quad \text{then} \quad \dot{\alpha} = \omega_z, \quad \alpha = \omega_z t + \alpha_0$$

If we set $\omega_z = \dot{\alpha} = 0$,

$$m\ddot{X} + mR\cos\alpha\ddot{\gamma} = 0$$
$$m\ddot{Y} - mR\sin\alpha\ddot{\gamma} = 0$$
$$C - mgR\sin\alpha\sin\varphi = (I + mR^2)\ddot{\gamma}$$

The necessary conditions for rolling up are where $C \geq mgR\sin\alpha\sin\varphi$. If the acceleration of rolling of the robot is positive then the acceleration along the

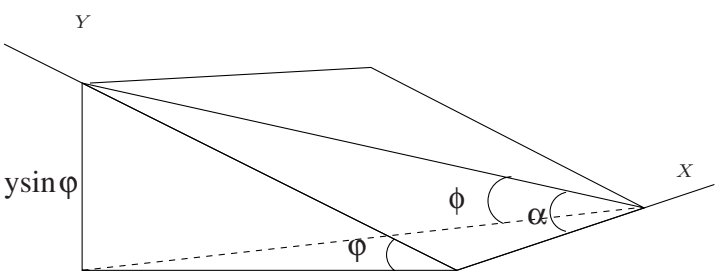


Fig. 2.8. Change orientation

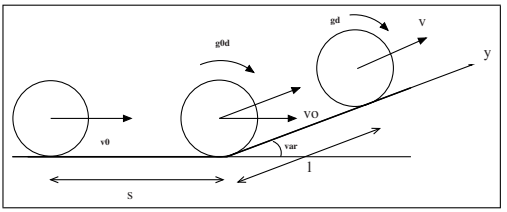


Fig. 2.9. Disk rolls on a plane

X and Y directions will become positive. In this way, we are sure that, in the direction of forward motion α_0, the robot can roll up on an inclined plane ϕ.

Method II : Change the initial velocities. Consider that a disk rolls along a straight line and then it hits an inclined surface. We assume that the robot does not bounce at the moment of hitting and the robot can remain perpendicular to the surface. We increase its angular velocity and linear velocity of the disk's center.

$$I\ddot{\gamma} = C - FR = C - maR, \qquad for \quad F = ma$$
$$\ddot{\gamma} = \frac{C}{mR^2 + I}, \qquad for \quad a = R\ddot{\gamma}$$

The angular velocity and the velocity of the disk are $\sqrt{\frac{2Cs}{IR+mR^3}}$ and $\sqrt{\frac{2CRs}{I+mR^2}}$ respectively. After it collides with the surface, based on the dynamic equations derived, the initial condition must be modified because $\dot{\gamma}_0, v_0$ are now non-zero in this case. Before the hitting on the inclined surface, we have

$$\ddot{y} = \frac{RC - mR^2 g \sin\varphi}{mR^2 + I}, \quad \ddot{\gamma} = \frac{C - mRg \sin\varphi}{mR^2 + I}$$

afterward, we have,

$$\dot{y} = \frac{RC - mR^2 g \sin\varphi}{mR^2 + I} t + v_0 \cos\varphi$$
$$\ddot{\gamma} = \frac{C - mRg \sin\varphi}{mR^2 + I} t + \dot{\gamma}_0$$

When $\dot{y} = \dot{y}_{min}$,

$$t = \frac{(\dot{y}_{min} - v_0 \cos\varphi)(mR^w + I)}{RC - mR^2 g \sin\varphi}$$
$$y = v_0 \cos\varphi t + \frac{1}{2}\Big(\frac{RC - mR^2 g \sin\varphi}{mR^2 + I}\Big)t^2$$
$$l = \frac{(mR^2 + I)(\dot{y}_{min}^2 - v_0 \cos\varphi)}{2(RC - mR^2 g \sin\varphi)}$$
$$(v_0 \cos\varphi)^2 = \dot{y}_{min}^2 - \frac{2l(RC - mR^2 g \sin\varphi)}{mR^2 + I}$$

If $\dot{y}_{min} = 0$, then

$$v_0^2 = \frac{2l(mR^2 g \sin\varphi - RC)}{\cos\varphi^2(mR^2 + I)}$$
$$l = \frac{v_0^2 \cos\varphi^2(mR^2 + I)}{2(mR^2 g \sin\varphi - RC)}$$

2.2.3 Simulation Study

Based on [112], we found that the property of Gyrover is similar to a rolling disk when its flywheel does not spin. If I_{xf} is zero, the dynamics of the robot is exactly the same as that of a rolling disk. The purpose of the high spinning flywheel is to provide a larger resistance to the rate of change of leaning, stabilizing the robot to balance. Now, we investigate how the high spinning flywheel provides a balancing effect for the robot when it climbs up on an inclined plane. In short, the robot, when its flywheel does not spin, cannot roll up an inclined plane i.e. a rolling disk cannot climb up an inclined plane. It is because the force components $mgRc\beta c\varphi$ and $-gmTs\alpha s\beta s\varphi$ are dominant on the roll dynamic. In the simulation, we assume that the single wheel robot is a rolling disk with $I_{xw} = I_{yw} = \frac{1}{2}mR^2$ and $I_{zw} = mR^2$. We use the following geometric/mass parameters from the real system throughout our simulations in Table 2.3.

Table 2.3. System parameters

Robot wheel:	$m = 15kg, R = 17cm, I_{xw} = 0.0289$
Flywheel:	$I_{xf} = 0.0063$
Static Friction :	$\mu_s = 1Nm/(rad/s)$
Sliding Friction :	$\mu_g = 0.1Nm/(rad/s)$
An Inclination angle:	$\varphi = 10^\circ$

The initial condition in the simulation study is

$$\dot{\alpha} = \dot{\beta} = 0\,\text{rad/s}, \dot{\gamma} = -15\,\text{rad/s}$$
$$\alpha = \gamma = 0^\circ, \beta = 80^\circ, X = Y = 0$$

Case I: Rolling up case

- Fig.2.10 shows the simulation of a rolling disk. It is noted that the lean angle $\dot{\beta}$ of the rolling disk decreases rapidly and the lean angle β becomes zero. This means the disk falls over, or, its inertial matrix will become singular, which violates the assumption of rolling without slipping. Besides, the steering rate $\dot{\alpha}$ rises up and down, and the trajectory of the rolling disk does not go up along the Y axis. The rolling disk fails to go up.
- For a single wheel robot, we investigated the system when $\dot{\gamma_a} = 1600\,\text{rpm} \neq 0$. It is clear that the spinning flywheel provides a large angular momentum. We stabilize the robot to the upright position $\beta = 90°, \delta\beta_{ref} = 0°$, such that the resulting roll-up trajectory is a straight line. The feedback gains are $k1 = -30$, $k2 = -3$ and $k3 = 3$ respectively. The simulation results are shown in Figure 2.11. It shows that the lean angle β of the robot exponentially converge to 90° and the steering rate $\dot{\alpha}$ exponentially converges to zero. The rolling speed of $\dot{\gamma}$ becomes -35 rad/s and the trajectory of the center of the robot oscillates at the beginning and then finally is restricted to follow a straight line path.

Case II: Rolling down case

- Fig.2.12 shows that a rolling disk falls down very rapidly from stationary. It is because 5° turns in the lean angle from the vertical direction makes the disk fall down.
- Fig.2.13 shows the single wheel robot when rolling down. We select the feedback gains which are $k1 = 30$, $k2 = -3$ and $k3 = -3$ respectively. It is similar to the rolling up case. This implies that the single wheel robot is balanced along the vertical position by the tilting of the flywheel.

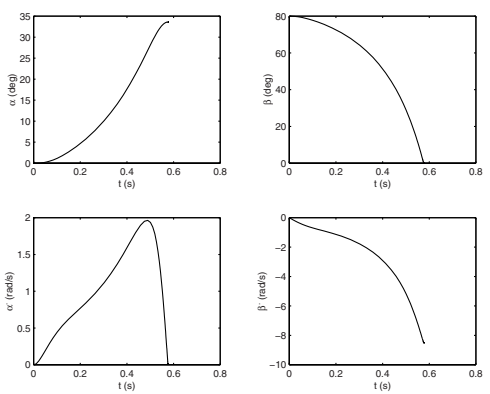


Fig. 2.10. Rolling up of a disk

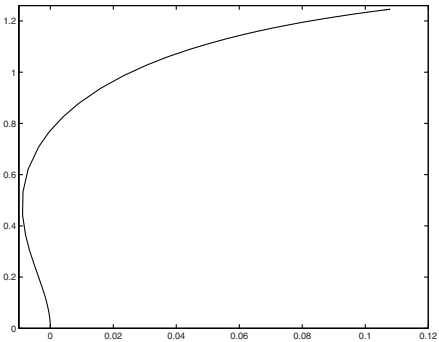


Fig. 2.10. Continued

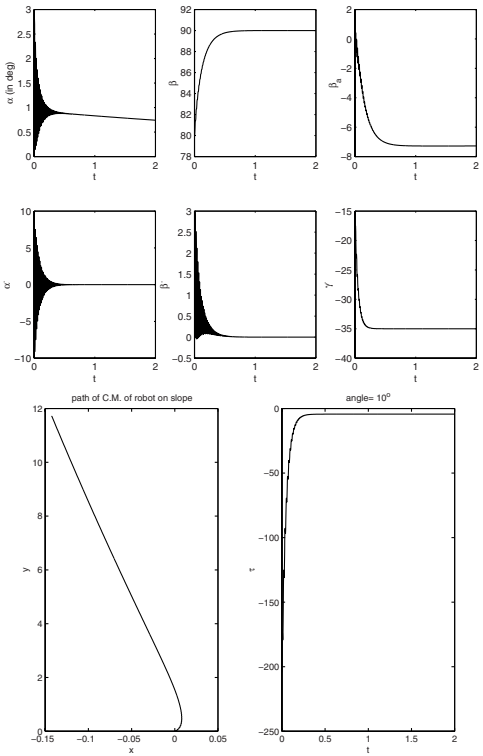


Fig. 2.11. Rolling up of a single wheel robot

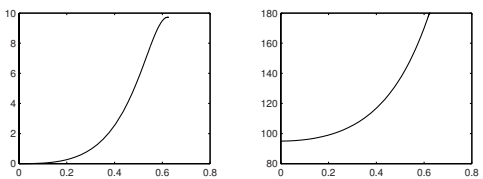


Fig. 2.12. Rolling down of a disk

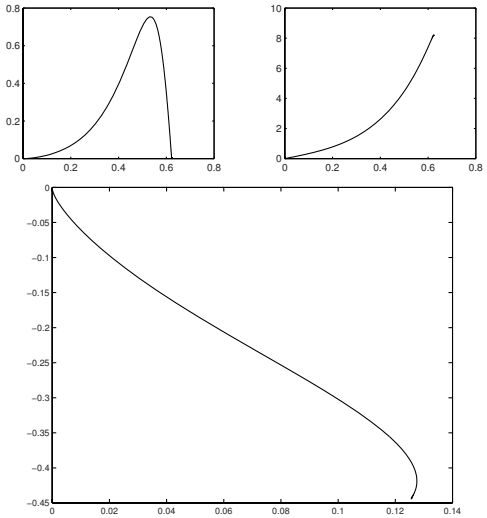


Fig. 2.12. Continued

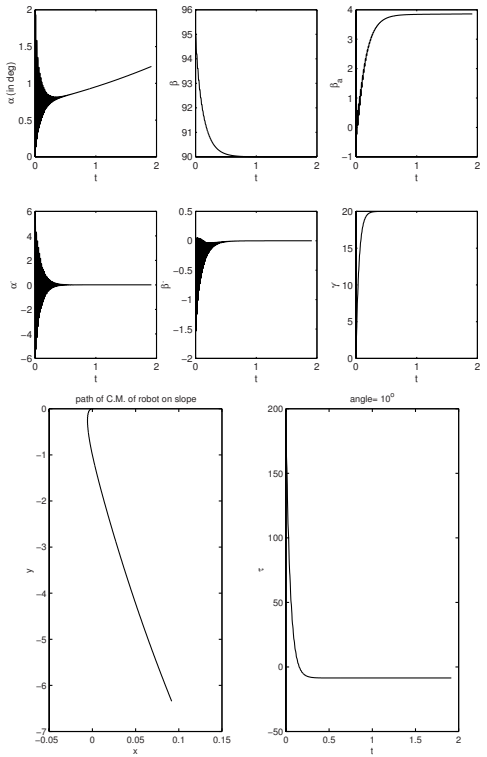


Fig. 2.13. Rolling down of a single wheel robot

3

Model-based Control

3.1 Linearized Model

3.1.1 Stabilization

Due to the inherent lateral instability, the first step in controlling the robot is stabilization. Fortunately, the lean angle of the robot can be controlled indirectly by tilting the flywheel. The single wheel robot steers by leaning at different angles. Therefore, it is necessary to design a controller that stabilizes the robot at any desired lean angle, so as to control the steering velocity.

On the other hand, for a typical unicycle, it should be noted that the longitudinal and lateral motions are highly coupled to each other, and it is not suitable to decompose the motions through a linearization method (Nakajima et al. [75] and Sheng [96]). However, because of the stabilizing effect of the flywheel, the effect of coupling/cross terms between the longitudinal and lateral motions of the robot become less significant for the single wheel robot. It is feasible to decouple them by linearizing the dynamic model, and then by designing a linear state feedback law to control the lean angle and rolling speed of the robot.

Linearized Model

We first linearize the system Eq. (2.32) about the vertical position. The definition of variables and the configuration of the linearized model are shown in Table 2.1 and Figure 3.1 respectively. In derivation of the linearized model, we make the folllowing assumptions: (1) the spinning rate $\dot{\gamma}_a$ is sufficiently large and remains constant, so that the terms $\dot{\gamma}_a\dot{\beta}, \dot{\gamma}_a\dot{\alpha}$ are sufficiently larger than the terms $\dot{\beta}\dot{\gamma}, \dot{\beta}\dot{\alpha}, \dot{\alpha}^2$. (2) the terms $\dot{\delta}_\beta, \dot{\delta}_{\beta_a}, \dot{\alpha}, \dot{\theta}$ are sufficiently small, (3) $\beta = 90^o + \delta_\beta, \dot{\gamma} = \Omega_o + \Omega, \beta_a = \delta_{\beta_a}$. The linearized model can be represented as

Y. Xu and Y. Ou: Control of Single Wheel Robots, STAR 20, pp. 33–71, 2005.

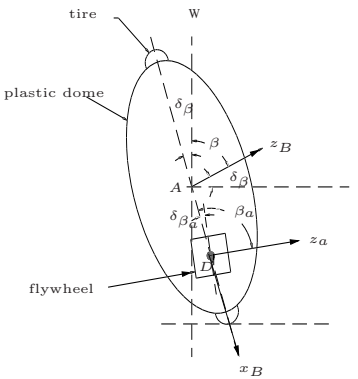


Fig. 3.1. The lateral description of Gyrover.

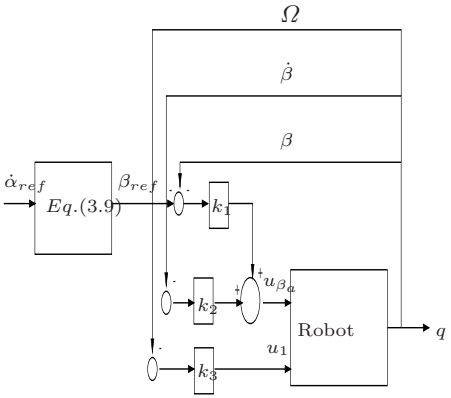


Fig. 3.2. Schematic of the control algorithms.

$$(I_{xxf} + I_{xxw})\ddot{\alpha} = 2(I_{xxw}\Omega_o + I_{xf}\dot{\gamma}_a)\dot{\delta}_\beta - \mu_s\dot{\alpha} + 2I_{xxf}\dot{\gamma}_a u_{\beta_a} \quad (3.1)$$

$$(I_{xxw} + mR^2)\ddot{\delta}_\beta = gmR\delta_\beta - (2I_{xxw} + mR^2)\Omega_o\dot{\alpha} - 2I_{xxf}\dot{\gamma}_a\dot{\alpha} \quad (3.2)$$

$$(2I_{xxw} + mR^2)\dot{\Omega} = -\mu_g\Omega + u_1 \quad (3.3)$$

Because Ω is independent of the roll and yaw dynamics (Eqs. (3.1) and (3.2)), we can decompose the longitudinal motion Eq. (3.3), and establish closed-loop control for controlling the angular velocity $\dot{\gamma}$ to the nominal value $\Omega_\circ$. The remaining yaw and roll dynamics form a state equation,

$$\dot{x} = Ax + Gu, \quad (3.4)$$

where $x = \left[\delta_\beta, \dot{\alpha}, \dot{\delta}_\beta\right]^T$,

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & a_{22} & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix}, G = \begin{bmatrix} 0 \\ b_2 \\ 0 \end{bmatrix}$$

where $a_{22}, ..., a_{32}$ and b_2 are derived from Eqs. (3.1) and (3.2).

Let $\mathcal{C}(A, G)$ be the controllability matrix of the system Eq. (3.4). Then

$$\begin{aligned} |\mathcal{C}(A,G)| &= |[\ G \mid AG \mid A^2G\ ]| \\ &= a_{32}^2 b_2^3 \\ &= (\frac{(2I_{xxw} + mR^2)\Omega_o + 2I_{xxf}\dot{\gamma}_a}{I_{xxw} + mR^2})^2 \times (\frac{2I_{xxf}\dot{\gamma}_a}{I_{xxf} + I_{xxw}})^3 \end{aligned} \quad (3.5)$$

From Eq. (3.5), the system is controllable if $\dot{\gamma}_a \neq 0$, for the reason that if the spinning rate of the flywheel is equal to zero, the flywheel does not provide an additional force for balancing the robot.

If we consider the steering velocity $\dot{\alpha}$ as the output of Eq. (3.4), then the transfer function becomes

$$\begin{aligned} H(s) &= c(sI - A)^{-1}b \\ &= \frac{b_2(-a_{31} + s^2)}{s^3 - a_{22}s^2 - (a_{31} + a_{23}a_{32})s + a_{22}a_{31}} \end{aligned} \tag{3.6}$$

where $c = [0, 1, 0]$ and $b = G = [0, b_2, 0]^T$. Note that there is a zero of $H(s)$ on the right-half plane, therefore it is a non-minimum phase. This means that the robot will fall if we control the steering rate $\dot{\alpha}$ only.

Linear State Feedback

Based on the system Eq. (3.4), if we stabilize the robot to a desired lean angle $\delta_{\beta ref}$ such that

$$\delta_\beta = \delta_{\beta ref}, \;\; \dot{\alpha} = \dot{\alpha}_s, \;\; \dot{\delta}_\beta = 0 \; rad/s, \;\; \ddot{\delta}_\beta = \ddot{\alpha} = \dot{\Omega} = 0 \; rad/s^2, \tag{3.7}$$

then the linear state feedback can be designed as

$$u_{\beta_a} = -k_1(\delta_\beta - \delta_{\beta ref}) - k_2\dot{\delta}_\beta - k_3(\dot{\alpha} - \dot{\alpha}_s). \tag{3.8}$$

where k_1, k_2, and k_3 are feedback gains and $\dot{\alpha}_s$ is the equilibrium steering rate under the conditions Eq. (3.7), that is

$$\dot{\alpha}_s = \frac{gmR\delta_{\beta ref}}{(2I_{xxw} + mR^2)\Omega_o + 2I_{xxf}\dot{\gamma}_a} \tag{3.9}$$

In order to ensure an asymptotic stability of the system Eq. (3.4), the necessary conditions of the feedback gains are

$$k_1 < 0, k_2 < 0, k_3 > 0.$$

Asymptotic stability ensures that all eigenvalues of the closed loop system have negative real parts. Note that for the given nominal rolling speed $\Omega_\circ$, the equilibrium steering rate $\dot{\alpha}_s$ corresponds to one, and only one lean angle $\delta_{\beta ref}$. Thus $\delta_{\beta ref}$ and $\dot{\alpha}_s$ can not be selected independently.

To allow the robot to track a desired path on the ground, we must be able to control its steering and linear velocities. For tracking the desired steering velocity $\dot{\alpha}_{ref}$, we propose to track a roll angle $\delta_{\beta ref}$ in which the equilibrium steering velocity $\dot{\alpha}_s$ will be equal to the desired one $\dot{\alpha}_{ref}$, according to Eq. (3.9). The schematic diagram for controlling the steering rate is shown in Figure 3.2.

3.1.2 Path Following Control

The kinematic constraints of a typical mobile robot with a steering front wheel can be written as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \upsilon \\ \omega \end{bmatrix} \tag{3.10}$$

where (x, y) is the position of the center of the vehicles, θ is the orientation, and (υ, ω) are the linear and steering velocities of the typical mobile robot respectively. A number of trajectory tracking methods have been proposed for the typical mobile robot [58], [87], [49], [89], [99]. However, because the single wheel robot is not assumed to be in a vertical position, the constraints described in Eq. (3.10) also depend on the lean angle and lean rate in this case. Furthermore, for the typical mobile robot, the steering velocity ω can be directly generated by turning the front wheel. The single wheel robot steers by leaning itself to a predefined angle. Therefore, the main difficulty in solving the path following problem of the single wheel robot is that we must not only control the position (x, y) and the orientation θ using two control inputs (υ, ω), but also control the lean angle β within a stable region to prevent the robot from falling.

Here, we propose an approach to the path following problem based on a geometrical notion in controlling the path curvature. We redefine the system configuration of the robot based on the geometrical notion and characteristics of the nonholonomic motion.

Robot Configuration

In the previous sections, we considered the center of mass (X_c, Y_c) to be the position of the robot. However, for path following where the robot needs to track a desired path on the ground, it is better to use the point of contact a on the ground to describe the position of the robot, instead of the center of mass C (Figure 2.1). Let (x_a, y_a) be the coordinates of the contact point a on the locus that coincides with a point of contact p of the robot. x_a and y_a can be expressed as

$$\begin{bmatrix} x_a \\ y_a \end{bmatrix} = \begin{bmatrix} X_c - RS_\alpha C_\beta \\ Y_c + RC_\alpha C_\beta \end{bmatrix} \tag{3.11}$$

Differentiating Eq. (3.11) with respect to time, velocity constraints at the contact point a are found to be

$$\begin{bmatrix} \dot{x}_a \\ \dot{y}_a \end{bmatrix} = \begin{bmatrix} \upsilon_a C_\alpha \\ \upsilon_a S_\alpha \end{bmatrix} \tag{3.12}$$

where υ_a is the contact point velocity,

$$\upsilon_a = R\dot{\gamma} \tag{3.13}$$

Note that Eq. (3.12) is independent of the lean angle β and is identical to the rolling constraints of a typical mobile robot, i.e. there is no drift term in Eq. (3.12).

The status of the contact point of the robot can be described by an alternative set of configurations based upon [48],

$$q = (p, \varphi, \kappa) = ((x_a, y_a), \varphi, \kappa)\,, \tag{3.14}$$

where (x_a, y_a), φ and κ are the position, the heading orientation and the path curvature of the robot with respect to the inertial frame, respectively. The new configuration of the robot is shown in Figure 3.3. The heading orientation φ is measured about the X-axis while the yaw angle α is measured about the Y-axis. Then

$$\varphi = \alpha - \frac{\pi}{2}, \;\; \dot{\varphi} = \dot{\alpha}. \tag{3.15}$$

Normally, we control the kinematic system of the robot using the steering and linear velocities to track a desired path. However, we can only indirectly control the steering velocity of a robot by changing its lean angle. Because the motion of the contact point a undergoes nonholonomic motion [48], its path curvature κ can be expressed as,

$$\kappa(t) = \frac{d\varphi(t)}{ds} = \frac{\dot{\varphi}(t)}{v_a(t)} = \frac{\dot{\alpha}(t)}{v_a(t)} = \frac{1}{r(t)} \tag{3.16}$$

where $r(t)$, s and $\dot{\varphi}(t)$ are the radius of the curvature from the center of rotation c to the contact point a, the path length, and the rotational velocity of the robot, respectively (Figure 3.3). If the center c of the rotation is at infinity, the robot is moving in a straight line and the path curvature and steering velocity are zero. From Eq. (3.16), for the given linear velocity v_a, we can control the path curvature κ by controlling the steering velocity $\dot{\alpha}$.

Line Following

We designed a line following controller for the robot to track a desired straight line. It is divided into two parts: (1) the velocity control law and (2) the torque control law. Using the velocity control law, the velocity inputs of the robot can be designed based on the error between the current configurations of the robot and the given trajectory. Assuming that the linear velocity of the robot is fixed/controlled to a nominal value, we consider the steering velocity as the only velocity input for the robot. Using the torque control law, we can design the tilt torque of the flywheel in order to lean the robot to a predefined angle which corresponds to the velocity input (steering velocity) obtained from the velocity control law.

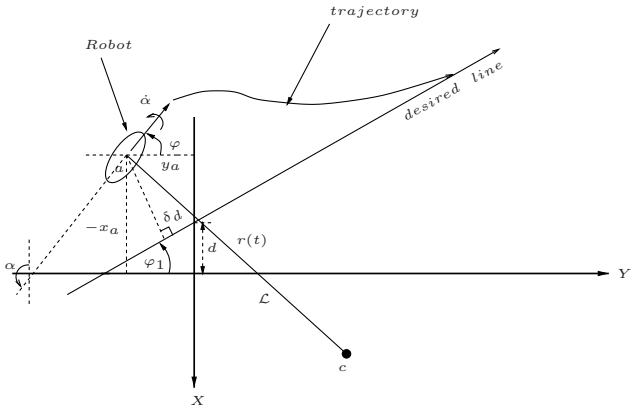


Fig. 3.3. Principle of line following.(top view)

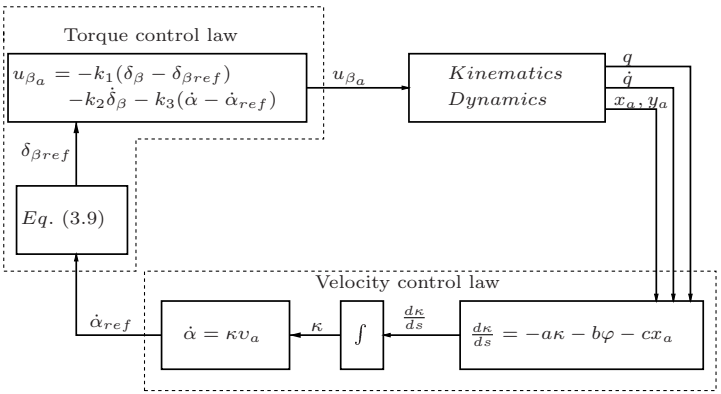


Fig. 3.4. Schematic of the control algorithm for the Y-axis.

Velocity control law

Unlike some typical control methods for the nonholonomic system, we used the path curvature as a variable to describe the path tracking of the robot with nonholonomic constraints. Therefore, based on [48], we consider the derivative of the path curvature as the velocity control law and then express the derivative of the path curvature ($\frac{d\kappa}{ds}$) with respect to the path length s as :

$$\frac{d\kappa}{ds} = -a\kappa - b(\varphi - \varphi_1) - c\delta_d, \tag{3.17}$$

where a, b and c are positive constants, φ_1 and δ_d are the direction of the desired line and the perpendicular distance between the robot and the desired line respectively (Figure 3.3). Eq. (3.17) is called a steering function. If (a, b, c) are selected such that Eq. (3.17) is asymptotically stable, the continuity of

the path curvature can be ensured, and $\frac{d\kappa}{ds} \to 0$ as the path length s increases, i.e., $\delta_d \to 0$, $\kappa \to 0$ and $\varphi \to \varphi_1$ as s increases. Hence, the robot converges to the desired straight line asymptotically. In order to design the velocity input (steering velocity), we first find the path curvative feedback κ by integrating the Eq. (3.17) in each instant. Using the given linear velocity and the path curvature feedback, we can obtain the corresponding velocity input for the robot according to Eq. (3.16).

Convergence of the velocity control law

To achieve convergence of the control law Eq. (3.17), we must select a proper set of coefficients (a, b, c). We first transform the coordinate system such that the desired path becomes the Y-axis, i.e., $\delta_d = -x_a, \varphi_1 = 0$. Then we introduce a new time scale which is identical to the distance along the desired path y_a and represent variables in Eq. (3.17) (x_a, φ, κ and $\frac{d\kappa}{ds}$) in terms of the derivative of x_a with respect to y_a. ($\frac{d^n x_a}{dy_a^n}$)

$$\varphi = tan^{-1}(\frac{dx_a}{dy_a}) \tag{3.18}$$

$$\begin{aligned}\kappa = \frac{d\varphi}{ds} &= \frac{\frac{d\varphi}{dy_a}}{\frac{ds}{dy_a}} \\ &= \frac{\frac{d^2x_a}{dy_a^2}}{(1+(\frac{dx_a}{y_a})^2)^{\frac{3}{2}}}\end{aligned} \tag{3.19}$$

$$\begin{aligned}\frac{d\kappa}{ds} &= \frac{\frac{d\kappa}{dy_a}}{\frac{ds}{dy_a}}, \\ &= \frac{d^3x_a}{dy_a^3}(1+\frac{dx_a}{dy_a}^2)^{-2} - 3\frac{dx_a}{dy_a}\frac{d^2x_a}{dy_a^2}^2(1+\frac{dx_a}{dy_a}^2)^{-3}\end{aligned} \tag{3.20}$$

Note that if $x_a \to 0$, $\frac{dx_a}{dy_a} \to 0$, $\frac{d^2x_a}{dy_a^2} \to 0$ and $\frac{d^3x_a}{dy_a^3} \to 0$, then $\varphi \to 0$, $\kappa \to 0$ and $\frac{d\kappa}{ds} \to 0$. Hence, the robot converges to the Y-axis. In order to track the Y-axis, x_a and its derivative with respect to y_a must converge to zero. To this end, we define a system of the first order equation,

$$\frac{d\xi}{dy_a} = f(\xi), \tag{3.21}$$

where

$$\xi \equiv \begin{bmatrix}\xi_1 \\ \xi_2 \\ \xi_3\end{bmatrix} \equiv \begin{bmatrix}x_a \\ \frac{dx_a}{dy_a} \\ \frac{d^2x_a}{dy_a^2}\end{bmatrix} \tag{3.22}$$

If the system of Eq. (3.21) is asymptotically stable, x_a and its derivative with respect to y_a converge to zero asymptotically. Using the Jacobian linearization, the system of Eq. (3.21) becomes

$$\frac{d\xi}{dy_a} = A\xi, \tag{3.23}$$

where

$$A = \left[\frac{df}{d\xi}\right]_{\xi=0} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -c & -b & -a \end{bmatrix} \tag{3.24}$$

To ensure the asymptotic stability of the system Eq. (3.23), a, b and c must be larger than zero while $ab > c$. The solution for critically damped conditions is,

$$a = 3k, \quad b = 3k^2, \quad c = k^3. \tag{3.25}$$

where k is the gain of the velocity control law.

$$\sigma = \frac{1}{k} \tag{3.26}$$

where σ is called the smoothness of the velocity control law.

Torque control law

By the velocity control law, the steering velocity input is defined when the linear velocity is given. Because of the absence of the front steering wheel, the robot steers only by leaning itself to a predefined angle. Therefore, we propose to design the tilt input of the flywheel u_{β_a} for tracking a lean angle trajectory $\delta_\beta(t)$ that is corresponding to the given steering velocity. Assuming that we can control the lean angle β faster than the control of the steering velocity $\dot{\alpha}$, we may consider $\dot{\alpha}$ to be fixed in the β time scale.

Based on Eq. (3.9), we can control the steering velocity $\dot{\alpha}$ to the desired value where the robot must be stabilized to the corresponding lean angle $\delta_{\beta ref}$. The linear state feedback for stabilizing the robot to the desired lean angle is

$$u_{\beta_a} = -k_1(\delta_\beta - \delta_{\beta ref}(t)) - k_2\dot{\delta}_\beta \tag{3.27}$$

where δ_β and $\delta_{\beta ref}$ are the small pertubation of the lean angle from the vertical position and the reference lean angle respectively. Note that the steering velocity $\dot{\alpha}$ does not appear in Eq. (3.27) because if the robot stabilizes to the desired lean angle $\delta_{\beta ref}$, its steering rate must converge to an equilibrium value according to Eq. (3.9). Therefore, there is no need to include the steering velocity in the state feedback Eq. (3.27). The overall architecture of the control algorithm has been shown in Figure 3.4.

3.1.3 Control Simulation

We first show the simulation results of the robot to track a desired line $y = \tan 30^\circ x$, with

$$\varphi_1 = \frac{\pi}{6}, \quad \delta_d = \frac{x_a - \tan 30^\circ y_a}{\sec 30^\circ}. \tag{3.28}$$

The simulation results are shown in Figure 3.5. The initial conditions in this simulation are shown in Table 3.1. The decoupled dynamic model [124] is adopted through out the simulations.

Table 3.1. The initial conditions for the simulations with different initial heading angles

Simulation	X_a	Y_a	$\alpha(0)$	$\beta(0)$	$\kappa(0)$	$\beta_a(0)$	$\varphi(0)$
$S1$	$5[m]$	0	70°	110°	0	0	-20°
$S2$	$5[m]$	0	110°	130°	0	0	20°
$S3$	$5[m]$	0	210°	110°	0	0	120°

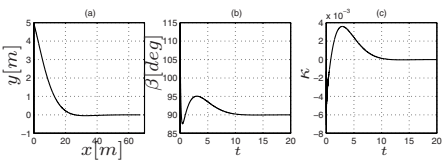


Fig. 3.5. The simulation results ($S1$) for following the Y-axis.

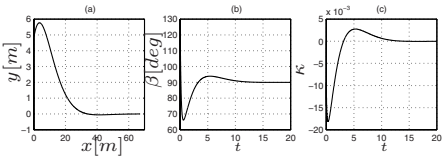


Fig. 3.6. The simulation results ($S2$) for following the Y-axis.

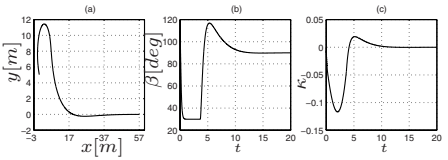


Fig. 3.7. The simulation results ($S3$) for following the Y-axis.

Figure 3.5 shows that the robot tracks the desired straight line without falling down ($\beta > 0^\circ$).

Effect of the initial heading angle

In this section, we show three simulation results ($S1$, $S2$ and $S3$) in which the robot tracks the Y-axis, under different initial conditions. The simulation

results and the initial conditions are shown in Figures 3.5, 3.6, 3.7 and Table 3.1, respectively. In these simulations, the rolling speed $\dot{\gamma}$ of the robot is controlled to a nominal rolling speed $\Omega_\circ = 30\ \ rad/s$, thus the contact point velocity v_a is also a constant. The smoothness is set to 30.

For simulation $S1$, the robot has a smaller path curvature than in simulation $S2$, because its initial heading angle $\varphi(0)$ in simulation $S1$ is less deviated from the Y-axis than in simulation $S2$. Furthermore, since the robot has larger path curvature in simulation $S2$, it results in sharper turns. To this end, the robot has to lean steeper to provide a sufficient steering velocity $\dot{\alpha}$ in simulation $S2$. The situation becomes more serious in simulation $S3$. Figure 3.7 shows the lean angle β is saturated in 5 sec. Because the initial heading angle $\varphi(0)$ is more than 90°, in order to ensure the curvature continuity while following the Y-axis, it should have the largest path curvature among them such that the lean angle β of the robot is the steepest, to provide a sufficient steering velocity. As we have already set up a limit for the state feedback controller, the lean angle can only be stabilized within $\beta \in (-60°, 60°)$. Otherwise, the controller will become saturated and the lean angle will be fixed at the limiting value. When the robot gradually approaches the Y-axis, the lean angle gradually increases until it reaches the vertical position (90°).

Effect of the rolling speed

In this section, we study the effect of the rolling speed of the robot $\dot{\gamma}$ on the path following controller. The simulation results with different rolling speeds are shown in Figures 3.8 and 3.9, both with the same initial conditions as in simulation $S1$. For $\dot{\gamma} = 30\ \ rad/s$, the robot converges to the Y-axis more rapidly than the other one, comparing Figures 3.8a and 3.9a. From Figures 3.8b and 3.9b, the change of the lean angle β of the robot with a lower rolling speed is more significant than that of the robot with a higher rolling speed. It is because, for the robot with a greater rolling speed, based on Eq. (3.16), a lesser steering velocity is required for tracking the same path curvature feedback κ. Thus, the change of the lean angle β decreases according to Eq. (3.9).

3.2 Nonlinear Model

Recently, there has been growing interest in the design of feedback control laws for nonholonomic systems [57]. Due to Brockett's theorem in [123], it is well known that a nonholonomic control system cannot be asymptotically stabilized to a resting configuration by smooth time-invariant control laws [18]. Despite this, several discontinuous or time-variant approaches have been proposed for stabilizing such systems in [18], [46], [57], [63] and [104]. The references above refer to systems with first-order nonholonomic constraints,

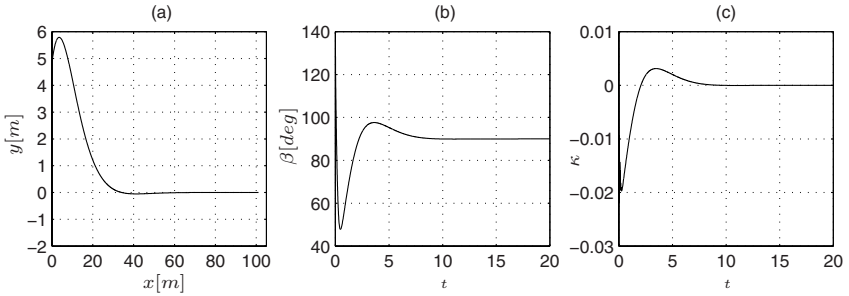


Fig. 3.8. The simulation results for following the Y-axis with $\dot{\gamma} = 10 \ \ rad/s$.

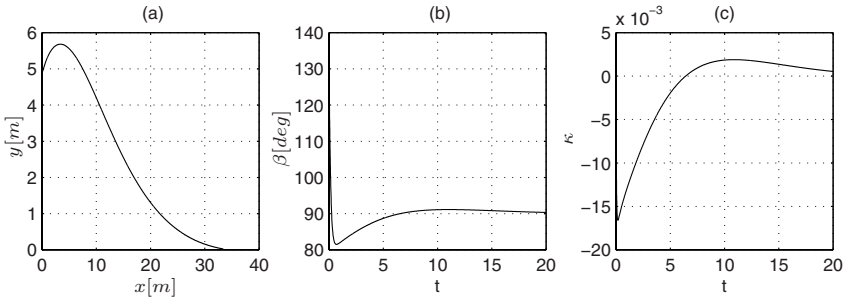


Fig. 3.9. The simulation results for following the Y-axis with $\dot{\gamma} = 30 \ \ rad/s$.

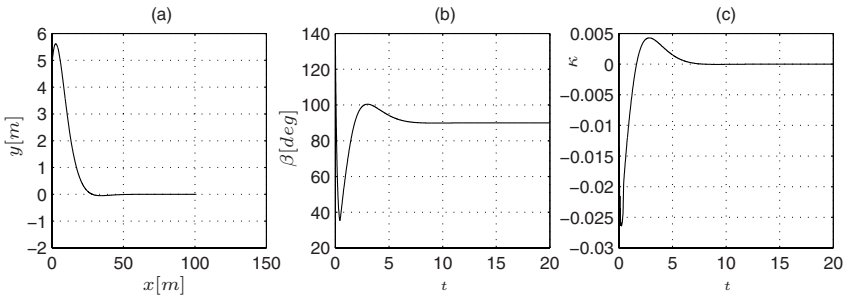


Fig. 3.10. The simulation results for following the Y-axis with $\sigma = 20$.

which can usually be expressed in terms of nonintegrable linear velocity relationships. However, there are another kind of mobile robot systems, which possess both first-order and second-order nonholonomic constraints, such as our robot – Gyrover, bicycles and motorcycles. Mobile robots have their inherent nonholonomic features, which can be described as first-order nonholonomic constraints among joint velocities and Cartesian space velocities. These constraints arise when robots roll on the ground without slipping. Because no actuators can be used directly for stabilization in the lateral direction, these

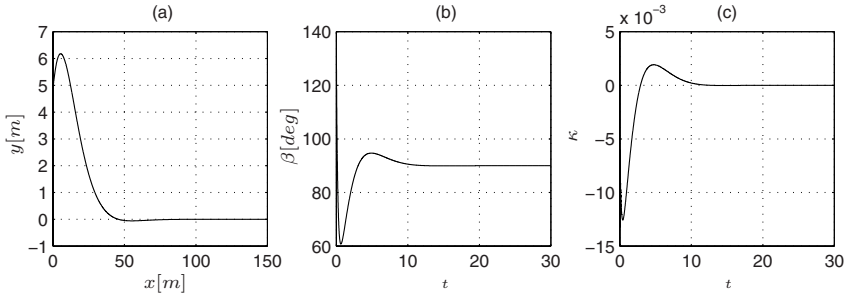


Fig. 3.11. The simulation results for following the Y-axis with $\sigma = 40$.

systems are underactuated nonlinear systems. This induces another nonholonomic constraint of robots. Thus, to compare with the above research work, our mobile robot systems – Gyrover, is more challenging to be controlled.

This work can provide some ideas for that class of problems. In this thesis, we want to control an underactuated mobile robot system – Gyrover. There are two control inputs: one is the steering torque (or steering rate) and the other is the driving torque (or driving speed). However, we have four independent generalized coordinates to control: (1) the lean angle, (2) the heading angle, (3) the Cartesian space X axis, (4) the Cartesian space Y axis.

Papers [6] and [7] assumed that the robot remained around the vertical position, which simplify this nonlinear system to a linear one. Therefore validation of the results can be limited. Some work had been about the tracking of a rolling disk, such as in [87], which assumed three control inputs in the direction of steering, leaning and rolling, where no unactuated joint and no second-order constraint is presented. In [34], the author simplified the bicycle dynamic model, and used velocities as control inputs to enable the lean angle (called "roll-angle" in that paper) to track trajectories, which have continuous differentials. However, the controller could not guarantee that the bicycle would not topple over, i.e. the lean angle was out of range, before convergence.

In this chapter, we focus on three control problems that have not yet been solved for Gyrover. The first one is the balance of the robot while standing. The second one is concerned with point-to-point control. The third one relates to following a straight line. These three problems are of significance in controlling a system with both first-order and second-order nonholonomic constraints.

Inertia Matrix and Nonholonomic Constraints

The kinematics and dynamics of Gyrover are different from those of unicycles, such as in [1]. The difference lies in the assumption that the unicycle always remains vertical. On the contrary, Gyrover can be considered as a rolling disk which is not constrained to the vertical position and is connected to a high

speed spinning flywheel. This model can serve well as a simplification for the study of a model of Gyrover.

Consider a disk rolling without slipping on a horizontal plane as shown in Figure 4.9. Let $\sum_o X, Y, Z$ and $\sum_c x, y, z$ be the inertial frame whose $x-y$ plane is anchored to the flat surface and the body coordinate frame whose origin is located at the center of the rolling disk, respectively. Let (X, Y, Z) be the Cartesian coordinates of the center of mass (c) with respect to the inertial frame $\sum_o$. Let A denote the point of contact on the disk. The configuration of the disk can be described by six generalized coordinates $(X, Y, Z, \alpha, \beta, \gamma)$, where α is the steering (precession) angle measured from X-axis to the contact line, $\beta \in (0, \pi)$ is the lean (nutation) angle measured from Z-axis to the z, and γ is the rolling angle. R is the radius of Gyrover. m, I_x, I_y and I_z represent the total mass and the moment of inertia of Gyrover.

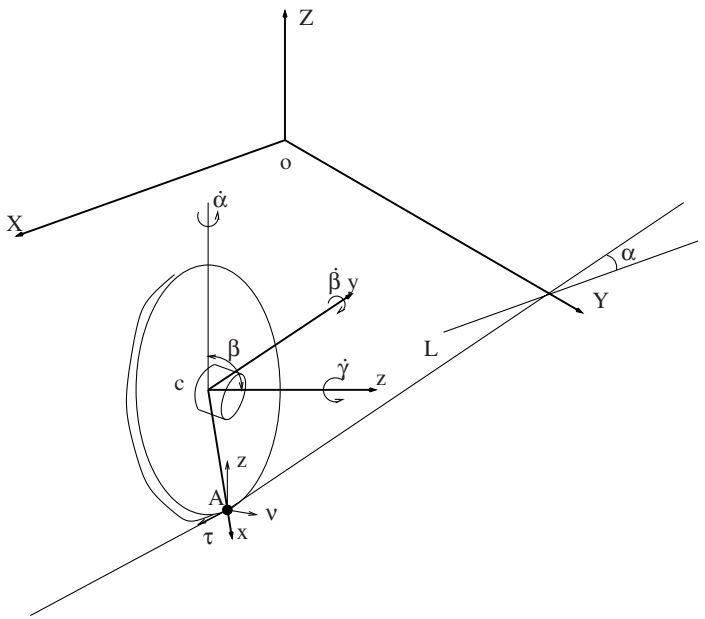


Fig. 3.12. System parameters of Gyrover's simplified model.

In the derivation of the model, we assume that the wheel rolls on the ground without slipping. Based on the previous derivation in [112] and letting $S_x := \sin(x)$, $C_x := \cos(x)$, the dynamic model is

$$M(q)\ddot{q} = N(q, \dot{q}) + Bu \tag{3.29}$$

$$\begin{cases} \dot{X} = R(\dot{\gamma}C_\alpha + \dot{\alpha}C_\alpha C_\beta - \dot{\beta}S_\alpha S_\beta) \\ \dot{Y} = R(\dot{\gamma}S_\alpha + \dot{\alpha}S_\alpha C_\beta + \dot{\beta}C_\alpha S_\beta) \end{cases} \tag{3.30}$$

where $q = [\alpha, \beta, \gamma]^T, N = [N_1, N_2, N_3]^T$

$$M = \begin{bmatrix} M_{11} & 0 & M_{13} \\ 0 & M_{22} & 0 \\ M_{31} & 0 & M_{33} \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}^T, u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\begin{aligned} M_{11} &= I_x S_\beta^2 + (2I_x + mR^2)C_\beta^2 \\ M_{13} &= (2I_x + mR^2)C_\beta \\ M_{22} &= I_x + mR^2 \\ M_{31} &= M_{13} \\ M_{33} &= 2I_x + mR^2 \\ N_1 &= (I_x + mR^2)S_{2\beta}\dot{\alpha}\dot{\beta} + 2I_x S_\beta \dot{\beta}\dot{\gamma} \\ N_2 &= -mgRC_\beta - (I_x + mR^2)C_\beta S_\beta \dot{\alpha}^2 - (2I_x + mR^2)S_\beta \dot{\alpha}\dot{\gamma} \\ N_3 &= 2(I_x + mR^2)S_\beta \dot{\alpha}\dot{\beta}, \end{aligned}$$

where (X, Y, Z) is the robot's center of mass coordinate with respect to the inertial frame as shown in Figure 4.9. $M(q) \in R^{3\times3}$ and $N(q, \dot{q}) \in R^{3\times1}$ are the inertial matrix and nonlinear terms, respectively. Equations (3.29) and (3.30) form the dynamic model and first-order nonholonomic constraints in the form of velocity.

In order to simplify the control design, we will transform the inertia matrix of the model to a diagonal matrix and reduce the nonlinear terms of the dynamic model. We first intend to cancel some nonlinear terms by letting

$$\begin{cases} u_1 = -N_1 + u_3 \\ u_2 = -N_3 + u_4. \end{cases} \tag{3.31}$$

Substituting Equation (3.31) into Equation (3.29) yields

$$\begin{cases} M_{11}\ddot{\alpha} + M_{13}\ddot{\gamma} = u_3 \\ M_{22}\ddot{\beta} \qquad\quad = -mgRC_\beta - (I_x + mR^2)C_\beta S_\beta \dot{\alpha}^2 - (2I_x + mR^2)S_\beta \dot{\alpha}\dot{\gamma} \\ M_{31}\ddot{\alpha} + M_{33}\ddot{\gamma} = u_4 \end{cases} \tag{3.32}$$

From the first and third equations in Equation (3.32) solving with respect to $\ddot{\alpha}$ and $\ddot{\gamma}$ one obtains

$$M_\rho = M_{11}M_{33} - M_{13}^2.$$

Due to its "shell" structure, even when Gyrover topples over, the lean angle is not zero, so that $S_\beta^2 > 0$ and $M_\rho > 0$ for all t. Then, we let

$$\begin{cases} u_5 = (M_{33}/M_\rho)u_3 - (M_{13}/M_\rho)u_4 \\ u_6 = -(M_{13}/M_\rho)u_3 + (M_{11}/M_\rho)u_4 \end{cases}$$

and obtain

$$\begin{cases} \ddot{\alpha} & = u_5 \\ M_{22}\ddot{\beta} & = -mgRC_\beta - (I_x + mR^2)C_\beta S_\beta \dot{\alpha}^2 - (2I_x + mR^2)S_\beta \dot{\alpha}\dot{\gamma} \\ \ddot{\gamma} & = u_6 \end{cases} \tag{3.33}$$

Equation (3.30) is nonintegrable, which is defined in [86]. Hence, it is the first-order nonholonomic constraint of the robot system. Moreover, we note that no control input is available for actuating directly on lean angle β. This forms a constraint in the form of accelerations. G. Oriolo in [80] proposed some necessary conditions for the partial integrability of second-order nonholonomic constraints, one of the conditions is that the gravitational term G_u is constant. In Equation (3.33), the gravitational term varies along β, thus it is a nonintegrable, second-order nonholonomic constraint.

3.2.1 Balance Control

Problem Statement

Our first control problem is to enable the robot to stand vertically, i.e., to stabilize the lean angle β to $\pi/2$ and $\dot{\beta}$, $\ddot{\beta}$, $\dot{\alpha}$ and $\dot{\gamma}$ to zero.

Based on the previous consideration, we are now able to specify more clearly the aforementioned closed loop steering problem in the following general terms.

Let the robot system defined in Equation (3.33) be initially moving in an undesired state and assume all the essential state variables are directly measurable. Then find a suitable (if any) state dependent control law $[u_5, u_6]^T$, which guarantees the states $[\beta - \pi/2, \dot{\beta}, \ddot{\beta}, \dot{\alpha}, \dot{\gamma}]$ to be asymptotically driven to the null limiting point $[0, 0, 0, 0, 0]^T$, while avoiding any attainment of the conditions of $\beta = 0$ (or $\beta = \pi$) in any finite time.

Balance Control

First, we let

$$\begin{cases} G_m & = mgR/M_{22} \\ I_m & = (I_x + mR^2)/M_{22} \\ J_m & = (2I_x + mR^2)/M_{22} \end{cases} \tag{3.34}$$

where G_m, I_m and J_m are positive scalar constant.

Then, the dynamic equations become

$$\begin{cases} \ddot{\alpha} = u_5 \\ \ddot{\beta} = -G_m C_\beta - I_m C_\beta S_\beta \dot{\alpha}^2 - J_m S_\beta \dot{\alpha}\dot{\gamma} \\ \ddot{\gamma} = u_6. \end{cases} \tag{3.35}$$

Letting $x^{(i)}$ be the i^{th} time derivative of x, then from Equation (3.35), we have

$$\beta^{(3)} = h_1(t)\dot{\beta} + h_2(t)u_5 + h_3(t)u_6 \tag{3.36}$$

where

$$\begin{cases} h_1(t) = G_m S_\beta - I_m C_{2\beta}\dot{\alpha}^2 - J_m C_\beta \dot{\alpha}\dot{\gamma} \\ h_2(t) = -I_m S_{2\beta}\dot{\alpha} - J_m S_\beta \dot{\gamma} \\ h_3(t) = -J_m S_\beta \dot{\alpha}. \end{cases}$$

Set $\beta(0) - \pi/2 = a,\ \dot{\beta}(0) = b,\ \ddot{\beta}(0) = c$, and a real number σ be

$$\sigma = |3a + 2b + c|/2 + |a + 2b + c|/2 + |a + b|/\sqrt{2}.$$

Proposition 1 Consider the system (3.35) with the feedback control laws u_5 and u_6,

$$u_5 = \begin{cases} -(\dot{\alpha} - \sqrt[4]{k_2 V}) \text{ if } \dot{\alpha}(0) > 0 \\ -(\dot{\alpha} + \sqrt[4]{k_2 V}) \text{ otherwise} \end{cases} \tag{3.37}$$

where V is defined in Equation (3.39) and k_2 is a positive number, which can be designed, and

$$u_6 = -(3(\beta - \pi/2) + 5\dot{\beta} + 3\ddot{\beta} + h_1(t)\dot{\beta} + h_2(t)u_5)/h_3(t) \tag{3.38}$$

where, $h_1(t), h_2(t), h_3(t)$ are defined as in Equation (3.36).

Any state $(\dot{\alpha}, \dot{\gamma}, \beta, \dot{\beta}, \ddot{\beta})$ starting from the domain D, which is defined as

$$\begin{aligned} D = \{&(\dot{\alpha}(0), \dot{\gamma}(0), \beta(0), \dot{\beta}(0), \ddot{\beta}(0)) | \dot{\alpha}(0) \neq 0, \\ &0 < \beta(0) < \pi,\ \sigma < \pi/2, \dot{\alpha}, \dot{\gamma}, \beta, \dot{\beta}, \ddot{\beta} \in R^1\} \end{aligned}$$

converges to the limiting point $[0, 0, \pi/2, 0, 0]^T$.

To present the proof, we need a lemma.

Lemma 1:

Consider a mechanical system with a constraint among a number of states $x_1(t), x_2(t), ..., x_n(t) \in R^1$. One of these state variables is uniquely determined by the other states; i.e., $x_n = f(x_1, x_2, ...x_{n-1})$. Let the limit of x_n exist as $t \to \infty$ and all the other states be asymptotically stabilizable to some real values. If all of the states are continuous and bounded, for all of t, then, x_n is also asymptotically stabilized to its limit, which is decided by the other states.

Proof:

First, since $x_n(t)$ is continuous and bounded, for all t, it is stable.

Second, because all of the other states are asymptotically stabilized to some values, thus

$$\lim_{t\to\infty} x_i = e_i \ exist \ \ i = 1, 2, ..., n-1.$$

Moreover, according to the property of limits, we have

$$\lim_{t\to\infty} x_n = \lim_{t\to\infty} f(x_1, x_2, ...x_{n-1}),$$

$$\lim_{t\to\infty} x_n = f(\lim_{t\to\infty} x_1, \lim_{t\to\infty} x_2, ..., \lim_{t\to\infty} x_{n-1}).$$

Because x_n is uniquely decided by the other states and has a limit as time goes to infinity, x_n will converge to the limit decided by the other states.

Then, we address the proof for **Proposition 1**.

Proof:

First, to prove the subsystem $\beta, \dot{\beta}, \ddot{\beta}$ asymptotically stabilized, we consider the following Lyapunov function candidate

$$V = (\beta - \pi/2)^2/2 + (\dot{\beta} + \beta - \pi/2)^2/2 + (\ddot{\beta} + 2(\dot{\beta} + \beta - \pi/2))^2/2. \quad (3.39)$$

We need to solve two more problems before we can prove the subsystem is asymptotically stable. One is whether $\dot{\alpha}$ is not zero in any finite time; the other is whether the controllers guarantee β is constrained in $(0, \pi)$. We will address the first problem later. We prove the second problem by replacing u_6 into Equation (3.35). We have

$$\beta^{(3)} = -(3(\beta - \pi/2) + 5\dot{\beta} + 3\ddot{\beta}). \quad (3.40)$$

By solving this linear differential equation, we obtain

$$\beta - \pi/2 = e^{-t}((3a + 2b + c) + \sqrt{2}(a + b)\sin(\sqrt{2}t) - (a + 2b + c)\cos(\sqrt{2}t))/2.$$

From $\sigma < \pi/2$, we know that $0 < \beta < \pi$ for all t.

Then, from Equation (3.39) the time derivative of V is

$$\dot{V} = (\beta - \pi/2) + (\dot{\beta} + \beta - \pi/2)(\ddot{\beta} + \dot{\beta}) + (\ddot{\beta} + 2(\dot{\beta} + \beta - \pi/2))(\beta^{(3)} + 2(\ddot{\beta} + \dot{\beta})).$$

By substituting Equation (3.40) into it, we have

$$\dot{V} = (\beta - \pi/2) + (\dot{\beta} + \beta - \pi/2)(\ddot{\beta} + \dot{\beta}) - (\ddot{\beta} + 2(\dot{\beta} + \beta - \pi/2))(3\ddot{\beta} + 3\dot{\beta} + 3(\beta - \pi/2)).$$

Then,

$$\dot{V} = -(\beta - \pi/2)^2 - (\dot{\beta} + \beta - \pi/2)^2 - (\ddot{\beta} + 2(\dot{\beta} + \beta - \pi/2))^2. \quad (3.41)$$

We can use the following Lyapunov function to prove $\dot{\alpha}$ converging to zero.

$$V_a = \sqrt{k_2 V}/4 + \dot{\alpha}^2/2.$$

From Equations (3.39) and (3.41), we have

$$\dot{V} = -2V. \quad (3.42)$$

The time derivative of V_α is

$$\dot{V}_a = \frac{\sqrt{k_2}\dot{V}}{8\sqrt{V}} + \dot{\alpha} u_\alpha.$$

By substituting Equations (3.42) and (3.37) into it,

$$\dot{V}_a = -\sqrt{k_2 V}/4 - (\dot{\alpha} \mp \sqrt[4]{k_2 V})\dot{\alpha}.$$

Then,

$$\dot{V}_a = -(\sqrt[4]{k_2 V}/2 \mp \dot{\alpha})^2. \tag{3.43}$$

Then we prove that the condition $\dot{\alpha} = 0$ can not be approached in any finite time. Without losing generality, we assume $\dot{\alpha}(0) > 0$ and let k_2 be 1.

If we let $V(0) = V_0$, from Equation (3.42), and by solving the differential equation, we obtain

$$V = e^{-2t} V_0.$$

By putting it into Equation (3.37), we have

$$\ddot{\alpha} = -(\dot{\alpha} - e^{-t/2} \sqrt[4]{V_0}).$$

By solving this differential equation, we obtain

$$\dot{\alpha} = e^{-t}\dot{\alpha}(0) + 2(e^{-t/2} - e^{-t}) \sqrt[4]{V_0}.$$

Thus, since $\dot{\alpha}(0) > 0$, $\dot{\alpha}$ can not reach zero in any finite time.

Since $\sin\beta$ and $\dot{\alpha}$ are not zero in any finite time, then $h_3(t)$ does not become zero for all t and from Equation (3.44), $\dot{\gamma}$ is continuous. We will prove that the limit of $\dot{\gamma}$ exists and is zero, as time goes to infinity.

From Lemma 1, if we asymptotically stabilize $\ddot{\beta}, \beta, \dot{\alpha}$ and guarantee that all of these states are continuous and bounded , $\dot{\gamma}$ will asymptotically converge to zero. Since $\dot{\alpha}$ will never be zero, there are two problems left. One is whether $\beta, \dot{\beta}, \ddot{\beta}$ can be stabilized and the other is whether we can guarantee $0 < \beta < \pi$ for all of t. From Equation (3.35), we have,

$$\dot{\gamma} = \frac{\ddot{\beta} + G_m C_\beta + I_m C_\beta S_\beta \dot{\alpha}^2}{-J_m S_\beta \dot{\alpha}}. \tag{3.44}$$

As V and $\dot{\alpha}$ reach very small values, $\sqrt{V}$ is the higher order small of $\sqrt[4]{V}$. For $|\beta - \pi/2| \leq \sqrt{V}$, $|\beta - \pi/2|$ is the higher order small of $\dot{\alpha}$, and so as to $\dot{\beta}$ and $\ddot{\beta}$. Using a Taylor series expansion, $\cos(\beta)$ is a higher order small of $\dot{\alpha}$. Thus, from Lemma 1 and the above equation, $\dot{\gamma}$ asymptotically converges to zero.

3.2.2 Position Control

Here, we propose a controller that drives the robot to the Cartesian space origin to study the point to point control problem. This is extremely important, for it serves as the basis for Cartesian space tracking.

Since the origin of the frame $XYZO(\sum o)$ in Figure 4.9 is fixed on the ground, for the purposes of tracking problems, it is more direct to use the point of contact A on the ground to describe the position of the robot, instead of the center of mass. Let (x_a, y_a) be the coordinates of the contact point A on

the ground that coincides with a point of contact P of the robot in Figure 4.9. x_a and y_a can be expressed as

$$\begin{bmatrix} x_a \\ y_a \end{bmatrix} = \begin{bmatrix} X - RS_\alpha C_\beta \\ Y - RC_\alpha C_\beta \end{bmatrix} \tag{3.45}$$

Differentiating Equation (3.45) with respect to time, we obtain,

$$\begin{cases} \dot{x_a} = R\dot{\gamma}C_\alpha \\ \dot{y_a} = R\dot{\gamma}S_\alpha \end{cases} \tag{3.46}$$

There are two kinds of input control commands for Gyrover: one set of control commands are torques and the other set of commands are velocities. The velocity commands u_α and u_γ control $\dot{\alpha}$ and $\dot{\gamma}$, respectively. Thus, Equations (3.30) and (3.33) transform into

$$\begin{cases} \dot{\alpha} &= u_\alpha \\ \ddot{\beta} &= -G_m C_\beta - I_m C_\beta S_\beta u_\alpha^2 - J_m S_\beta u_\alpha u_\gamma \\ \dot{\gamma} &= u_\gamma \\ \dot{x_a} &= Ru_\gamma C_\alpha \\ \dot{y_a} &= Ru_\gamma S_\alpha \end{cases} \tag{3.47}$$

where G_m, I_m and J_m are the same as in Equation (3.34).

Problem Statement

Let us consider the robot with respect to the inertial frame $XYZO(\sum o)$, as shown in Figure 3.13. By representing the Cartesian position of the robot in terms of its polar coordinates, which involves the error distance $e \geq 0$,

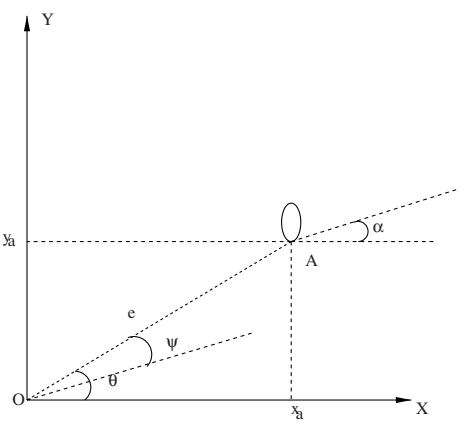


Fig. 3.13. Parameters in position control.

measured from A to O (the origin of the frame), Gyrover's orientation θ with respect to the inertial frame, and defining $\psi = \theta - \alpha$ as the angle measured between the vehicle principal axis and the distance vector e. When $e = 0$, there is no definition for θ and ψ. Then the following equations are obtained

$$\begin{aligned} \dot{e} &= Ru_\gamma C_\psi \\ \dot{\psi} &= -u_\alpha - S_\psi u_\gamma / e \end{aligned} \tag{3.48}$$

where $C_\psi := \cos(\psi)$, $S_\psi := \sin(\psi)$ and $e \neq 0$. Moreover, we define $\psi = 0$ and $\dot{\psi} = -u_\alpha$, if $e = 0$.

On the basis of the previous considerations, we are now ready to address the aforementioned closed loop steering problem in the following general terms.

Let the robot system be initially located at any non-zero distance from the inertial frame and assume that all state variables required are directly measurable. Then find a suitable, if any, state feedback control law $[u_\alpha, u_\gamma]^T$ which guarantees the state $[e, \beta - \pi/2, \dot{\beta}]$ to be asymptotically driven to the null point $[0, 0, 0]^T$, while avoiding any attainment of the conditions of $\beta = 0$ (or $\beta = \pi$) in a finite time.

Proposed Controller

Proposition 2 Consider the system (3.47) with the feedback control laws u_α and u_γ,

$$\begin{cases} u_\alpha = -k_3 Sgn(C_\psi) Sgn(\beta - \pi/2 + \dot{\beta}) \\ u_\gamma = -(k_4 e + u_k) Sgn(C_\psi) \end{cases} \tag{3.49}$$

where Sgn, K_3 and k_4 are defined in Equations (3.50) and (3.51), k_4 is a positive scalar constant and $k_4 < k_3 - 1$, which can be designed.

Any state $(e, \beta - \pi/2, \dot{\beta})$ starting from the domain D defined by

$$\begin{aligned} D = \{&(e(0), \beta(0) - \pi/2, \dot{\beta}(0))|\ e > 0,\ 0 < \beta < \pi, \\ &\sqrt{(\beta - \pi/2)^2 + (\beta - \pi/2 + \dot{\beta})^2/2} < \pi/2, \\ &e, \beta, \dot{\beta} \in R^1\} \end{aligned}$$

converges to the point $[0, 0, 0]^T$.

Proof: First, let $Sgn(.)$ be a sign function described as follows:

$$Sgn(x) = \begin{cases} 1, & if\ x \geq 0 \\ -1, & if\ x < 0 \end{cases} \tag{3.50}$$

Let $k_3 > 2$ be a positive scalar constant, which can be designed and should be less then $\dot{\alpha}_{mam}$. Let

$$\begin{aligned} f_1 &= \left| G_m C_\beta + I_m C_\beta S_\beta k_3^2 \right| \\ u_k &= (\left| 2(\beta - \pi/2 + \dot{\beta}) \right| + f_1)/(J_m S_\beta k_3). \end{aligned} \tag{3.51}$$

Set

$$\begin{aligned} V &= V_1 + V_2 \\ V_1 &= ((\beta - \pi/2)^2 + (\beta - \pi/2 + \dot{\beta})^2)/2 \\ V_2 &= e^2/2. \end{aligned} \tag{3.52}$$

The time derivative $\dot{V}$ is given by

$$\dot{V} = \dot{V}_1 + \dot{V}_2$$

where

$$\dot{V}_1 = 2(\beta - \pi/2)\dot{\beta} + (\beta - \pi/2 + \dot{\beta})(\dot{\beta} + \ddot{\beta})$$

Substituting Equations (3.51) and (3.49) into Equation (3.47), we obtain

$$\begin{aligned} (\beta - \pi/2 + \dot{\beta})\ddot{\beta} &= -(G_m C_\beta + I C_\beta S_\beta k_3^2)(\beta - \pi/2 + \dot{\beta}) \\ &\quad - \left|(G_m C_\beta + I_m C_\beta S_\beta k_3^2)(\beta - \pi/2 + \dot{\beta})\right| \\ &\quad - J_m S_\beta k_3 k_4 e \left|\beta - \pi/2 + \dot{\beta}\right| - 2(\beta - \pi/2 + \dot{\beta})^2 \\ &\leq -2(\beta - \pi/2 + \dot{\beta})^2 \end{aligned}$$

such that

$$\dot{V}_1 \leq -(\beta - \pi/2)^2 - \dot{\beta}^2 - (\beta - \pi/2 + \dot{\beta})^2$$

Thus V_1 is positive definite and $\dot{V}_1$ is negative definite. The remaining problem concerns why β will not reach 0 or π during the entire process. Since $|\beta - \pi/2| \leq \sqrt{V_1(0)} < \pi/2$ and V_1 is monotonically non increasing, they guarantee $0 < \beta < \pi$.

$$\begin{aligned} \dot{V}_2 &= e\dot{e} \\ &= -eR\,|C_\psi|\,(k_4 e + u_k) \\ &\leq -k_4 e^2 R\,|C_\psi| \end{aligned}$$

Since R, k_3, k_4, S_β and u_k are positive, $\dot{V}_2 \leq 0$. In fact, $\dot{V}_2$ is strictly negative, except when

$$\begin{aligned} e &= 0 \\ C_\psi &= 0 \end{aligned} \tag{3.53}$$

In these cases $\dot{V}_2 = 0$. Equation (3.53) presents two possible solutions for the system.

Moreover, $C_\psi = 0$ is not a solution, which is evident from Equation (3.48). By substituting u_α and u_γ into Equation (3.48), we have

$$\dot{\psi} = Sgn(C_\psi)(k_3 Sgn(\beta - \pi/2 + \dot{\beta}) + S_\psi(k_4 e + u_k)/e).$$

Because $\beta \to \pi/2$, $\dot{\beta} \to 0$ and $C_\beta \to 0$, u_k will vanish. Thus we obtain

$$\dot{\psi} = Sgn(C_\psi)(k_3 Sgn(\beta - \pi/2 + \dot{\beta}) + k_4 S_\psi).$$

Since $k_3 - 1 > k_4 > 0$, during any sampling period, $\dot{\psi}$ is not zero. Hence, any system trajectory starting from a set in $C_\psi = 0$ will not remain there. Thus, $e = 0$ is the only solution, according to the LaSalle Proposition for nonsmooth systems in [95]. Therefore, the proposition is proven.

3.2.3 Line Tracking Control

For a mobile robot, such as Gyrover, most traveling tasks can be realized by following connected segments of straight lines. For example, Figure 3.14 shows a mobile robot's travel path along a series of connected corridors.

How to realize the straight line following while keeping the robot vertical is our control problem. As depicted in Figure 3.15, where (x_2, y_2) is the coordinate of the second point, let us consider Gyrover initially positioned at a neighborhood of the origin (i.e., the start point) and standing almost vertically, about to take a straight line so as to approach the second point.

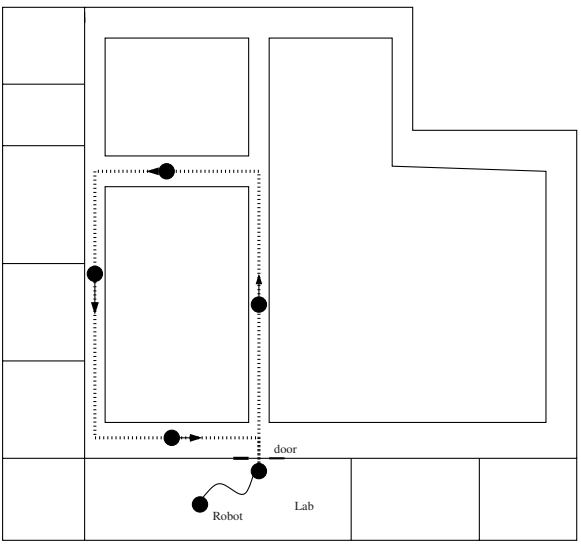


Fig. 3.14. The robot's path along connected corridors.

Let us define r, e and d, which are nonnegative, as the distances from A to the origin, the line and the second point, respectively. θ and ϕ are defined as in Figure 3.15. Then, we have the following equations

$$\begin{cases} e\dot{e} = rRu_\gamma S_{\phi-\alpha} S_{\phi-\theta} \\ d\dot{d} = pRu_\gamma \\ p = rC_{\theta-\alpha} - \sqrt{x_2^2 + y_2^2} C_{\phi-\alpha} \\ \quad = -dC_{\varphi-\alpha} \end{cases}$$

where $S_{\phi-\alpha} := \sin(\phi - \alpha)$, $S_{\phi-\theta} := \sin(\phi - \theta)$, $C_{\phi-\theta} := \cos(\phi - \theta)$, and $C_{\varphi-\alpha} := \cos(\varphi - \alpha)$.

On the basis of previous considerations, we are now ready to address the aforementioned closed loop line tracking problem in general terms.

Let the robot initially locate at some neighborhood of the origin and assume that all state variables be directly measurable; then find a suitable (if

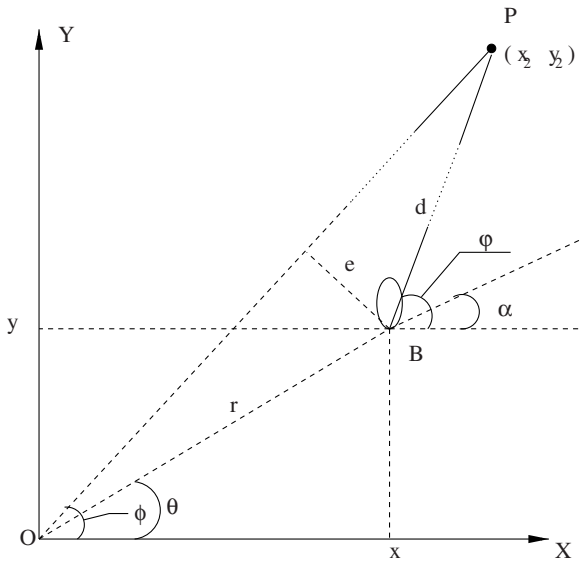


Fig. 3.15. The parameters in the line tracking problem.

any) feedback control law $[u_\alpha, u_\gamma]^T$ that guarantees the state $[\beta - \pi/2, \dot{\beta}, e, d]^T$ to be asymptotically driven to the point $[0, 0, 0, 0]^T$, while avoiding any attainment of the conditions of $\beta = 0$ (or $\beta = \pi$) in finite time.

Proposition 3 Consider the system (3.47) with the feedback control laws u_α and u_γ,

$$\begin{cases} u_\alpha = -k_3 Sgn(S_{\phi-\alpha} S_{\phi-\theta}) Sgn(\beta - \pi/2 + \dot{\beta}) \\ u_\gamma = -(f_2 + u_k) Sgn(S_{\phi-\alpha} S_{\phi-\theta}) \end{cases} \tag{3.54}$$

where f_2 is defined in Equation (3.56).

Any state, starting from $[\beta(0) - \pi/2, \dot{\beta}(0), e(0), d(0)]$ with $0 < \beta(0) < \pi$ and $e(0) > 0$, converges to the point $[0, 0, 0, 0]^T$.

Proof:

First, we introduce some definitions.

$Sgn(.)$, f_1 and u_k are defined as in Equations (3.50) and (3.51).

Let $\Theta(.)$ be a sign function described as follows:

$$\Theta(x) = \begin{cases} 1, \, if \; x \geq 0 \\ 0, \, if \; x < 0. \end{cases} \tag{3.55}$$

We let

$$f_2 = k_5 \Theta(p Sgn(S_{\phi-\alpha} S_{\phi-\theta})) \tag{3.56}$$

where k_5 is a positive scalar constant which can be designed and should be less than $\dot{\gamma}_{max}$.

Set

$$\begin{aligned} V &= V_1 + V_2 + V_3 \\ V_1 &= ((\beta - \pi/2)^2 + (\beta - \pi/2 + \dot{\beta})^2)/2 \\ V_2 &= e^2/2 \\ V_3 &= d^2/2 \end{aligned} \tag{3.57}$$

The time derivative $\dot{V}$ is given by

$$\dot{V} = \dot{V}_1 + \dot{V}_2 + \dot{V}_3$$

where

$$\begin{aligned} \dot{V}_1 &= (\beta - \pi/2)\dot{\beta} + (\beta - \pi/2 + \dot{\beta})(\dot{\beta} + \ddot{\beta}) \\ &\leq -(\beta - \pi/2)^2 - (\beta - \pi/2 + \dot{\beta})^2 \\ \dot{V}_2 &= e\dot{e} \\ &= -rR\,|S_{\phi-\alpha}S_{\phi-\theta}|\,(f_2 + u_k) \\ \dot{V}_3 &= d\dot{d} \\ &= pRu_\gamma. \end{aligned}$$

Since r, R, f_2 and u_k are nonnegative, $\dot{V}_2 \leq 0$. This means that the first term e is always non increasing in time and in the consequence. If $d = 0$ is maintained, e is also zero. Moreover, from Figure 3.15, $S_{\phi-\theta} = 0$ and $e = 0$ can be deduced from each other. $\dot{\alpha}$ will never be zero, for $\dot{\alpha} = u_\alpha$. Because ϕ is a constant value, it is trivial to know that e will not stop decreasing until $e = 0$.

V_1 is positive definite and $\dot{V}_1$ is negative definite. At the beginning β is near vertical, so that during the entire process, $\beta \approx \pi/2$ is sustained. That means $\cos\beta \to 0$ and $u_k \doteq 0$. By omitting them,

$$\dot{V}_3 = -k_5 pSgn(S_{\phi-\alpha}S_{\phi-\theta})\Theta(pSgn(S_{\phi-\alpha}S_{\phi-\theta})).$$

thus, $\dot{V}_3$ is negative semi-definite. As with the previous process, it is trivial to prove that the only solution of the system, for $V_3 = 0$ is $d = 0$. Thus, we prove the proposition.

3.2.4 Simulation Study

The following simulation study is based on robot system parameters as shown in Table 3.2.

Table 3.2. Physical parameters.

$m\ [kg]$	$R\ [m]$	$g\ [m/s^2]$	$I_x\ [kgm^2]$	I_y	I_z
8.05	0.17	9.8	0.116	0.116	0.232

Balance Control

With regard to balance control, the initial condition seems to be too narrow for the constraint, $\sigma < \pi/2$. If we make a small modification to u_6 as follows, the controller can be used in a wider range of initial conditions.

$$u_6 = -((2+k_1)(\beta - \pi/2) + (3+2k_1)\dot{\beta} + (2+k_1)\ddot{\beta} + h_1(t)\dot{\beta} + h_2(t)u_5)/h_3(t)$$

where k_1 is a positive number, which can be designed. To prove the subsystem $\beta, \dot{\beta}, \ddot{\beta}$ is asymptotically stabilized, we consider the following Lyapunov function candidate

$$V^* = (\beta - \pi/2)^2/2 + (\dot{\beta} + \beta - \pi/2)^2/2 + (\ddot{\beta} + (1+k_1)(\dot{\beta} + \beta - \pi/2))^2/2.$$

Its derivative is

$$\dot{V}^* = -(\beta - \pi/2)^2 - k_1(\dot{\beta} + \beta - \pi/2)^2 - (\ddot{\beta} + (1+k_1)(\dot{\beta} + \beta - \pi/2))^2.$$

We completed two sets of simulations for balance control to compare the different effects of k_1. The initial parameters are shown in Table 3.3.

Table 3.3. Initial parameters in balance control.

β_0 $[rad]$	$\dot{\beta}_0[rad/s]$	$\dot{\alpha}_0$	$\dot{\gamma}_0$
$\pi/4$	3	8	-8.660381

In the first simulation, $k_1 = 10$, the target and other gain parameters are shown in Table 3.4.

Table 3.4. Target and gain parameters in balance control I.

β_d $[rad]$	$\dot{\beta}_d$ $[rad/s]$	$\dot{\alpha}_d$	$\dot{\gamma}_d$	k_1	k_2
$\pi/2$	0	0	0	10	1

The time span of the simulation is from 0 to 16.0s. The results are shown in Figures 3.16, 3.17, 3.18 and 3.19 .

In the second simulation, except for $k_1 = 1$, all other parameters are the same as in the first simulation.

The time span of the simulation is from 0 to 16.0s. The results are shown in Figures 3.20, 3.21, 3.22 and 3.23 .

Even though it is possible that the states in simulation II will converge, the control process in simulation I is faster and smoother.

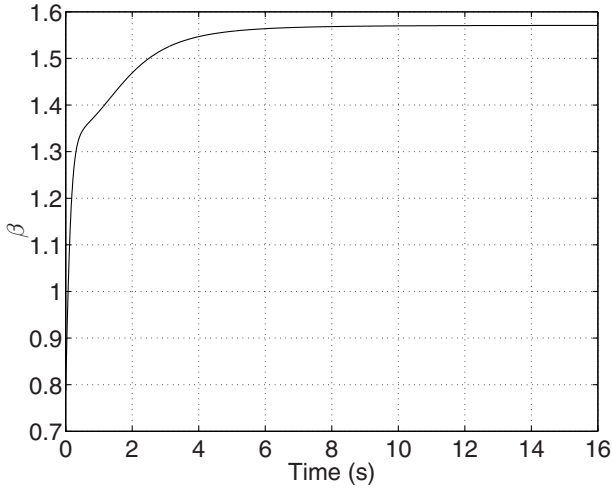


Fig. 3.16. Leaning angle β in balance control.

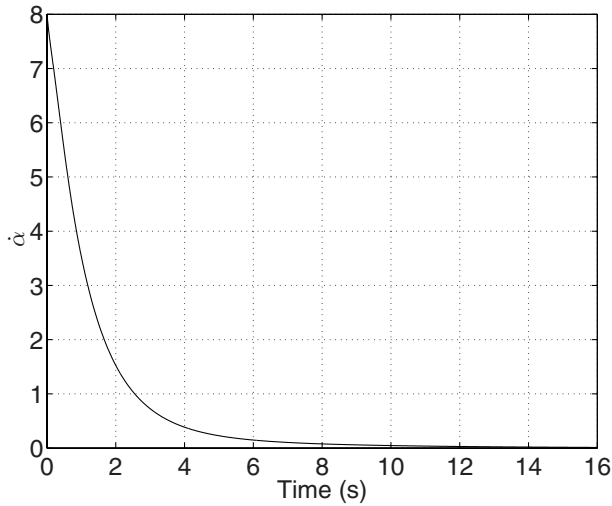


Fig. 3.17. Precession angle velocity $\dot{\alpha}$ in balance control.

Position Control

We have set the initial values of the simulation in Table 3.5.

The target parameters are shown in Table 3.6.

The time span of the simulation is from 0 to 16s. The results are shown in Figures 3.24, 3.25, 3.26 and 3.27.

This simulation demonstrates that the proposed point to point controller is correct.

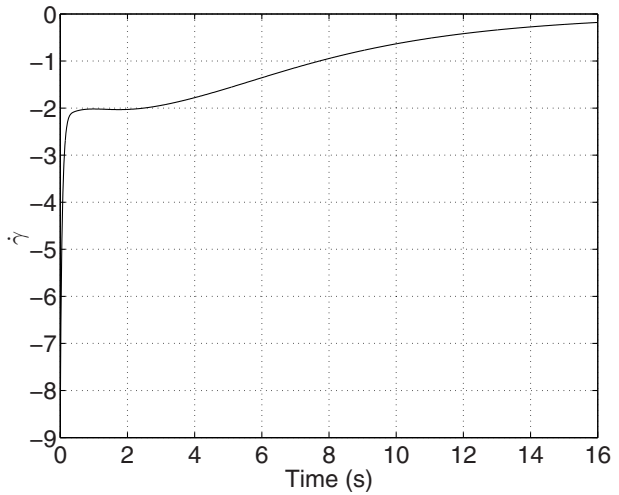


Fig. 3.18. Driving speed $\dot{\gamma}$ in balance control.

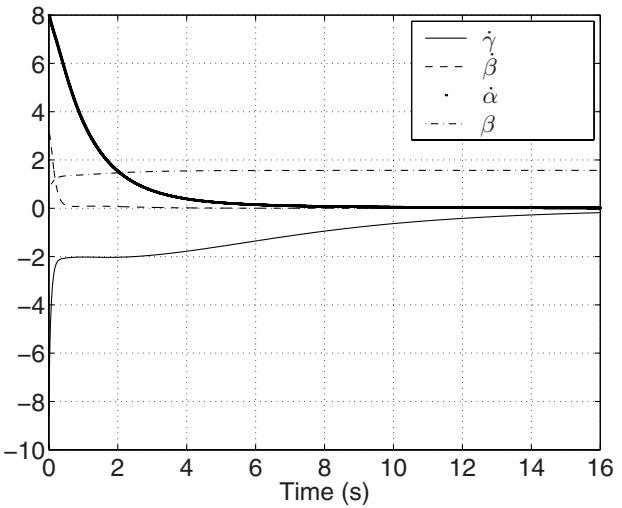


Fig. 3.19. Parameters of $\dot{\alpha}, \beta, \dot{\beta}, \dot{\gamma}$ in balance control.

Line Tracking Control

The aim of the simulation is to track a straight line so as to demonstrate the efficiency of the proposed line tracking controller. From the neighborhood of an origin, the robot is driven along a straight line to reach the point(3,4).

The initial values of the simulation are shown in Table 3.7.

The target parameters are shown in Table 3.8.

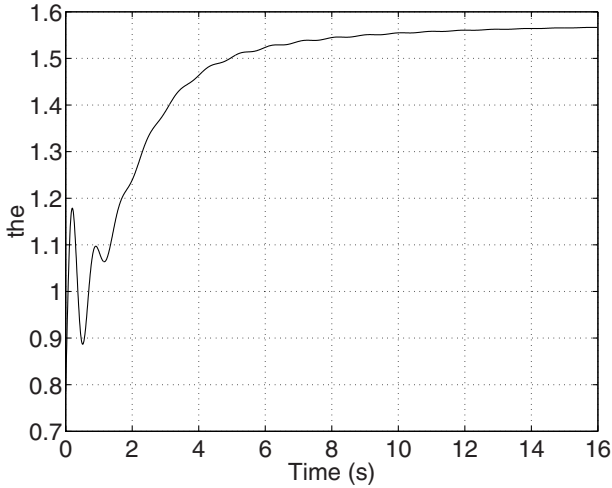


Fig. 3.20. Leaning angle β in balance control II.

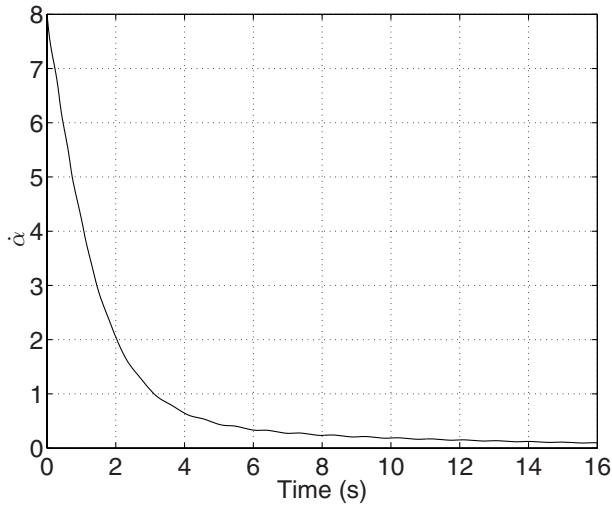


Fig. 3.21. Precession angle velocity $\dot{\alpha}$ in balance control II.

The time span of the simulation is from 0 to 32s. The results are shown in Figures 3.28, 3.29, 3.30 and 3.31.

Here, large values of k_3 and k_5 are better, however, they should be less than the $\dot{\alpha}_{max}$ and $\dot{\gamma}_{max}$, respectively.

Despite this, there are some drawbacks in the control process. The first is the system states exhibit highly oscillatory behavior and the second is that control inputs sometimes need to switch too sharply and fast. Both will cause difficulties for real time control and result in worse performance. To solve

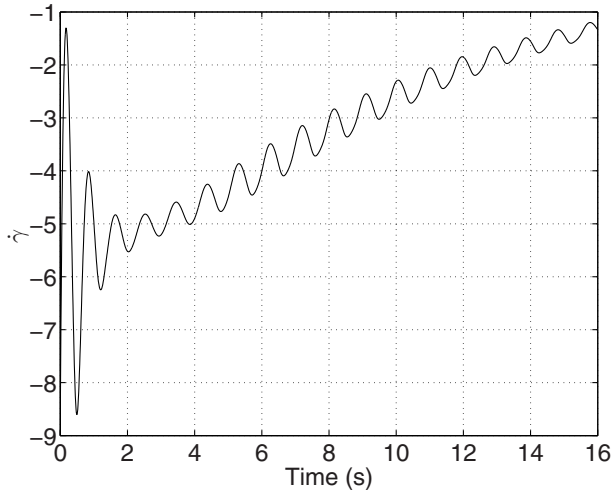


Fig. 3.22. Driving speed $\dot{\gamma}$ in balance control II.

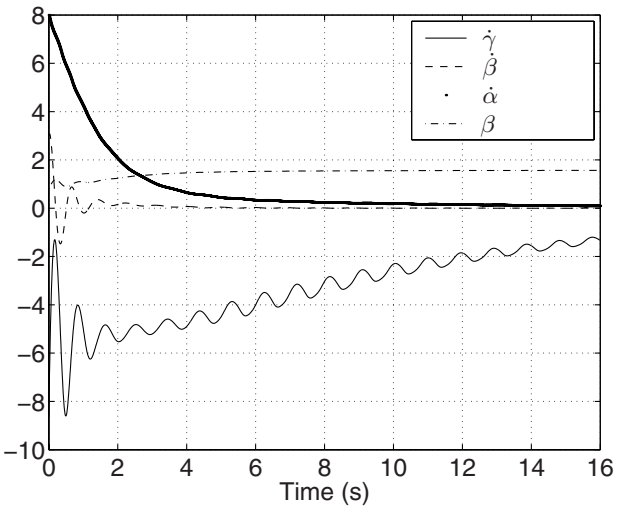


Fig. 3.23. Parameters of $\dot{\alpha}, \beta, \dot{\beta}, \dot{\gamma}$ in balance control II.

these problems, we propose to replace the sign functions in the controllers with tanh functions.

Definition 3 $Tanh(.)$ is a bipolar function described as follows:

$$Tanh(x) = \frac{1 - e^{-k_6 x}}{1 + e^{-k_6 x}} \tag{3.58}$$

where k_6 is a positive scalar constant, which can be designed. $Tanh(.)$ is shown as in Figure 3.32.

Table 3.5. Initial parameters in position control.

β_0 $[rad]$	$\dot{\beta}_0$ $[rad/s]$	$\dot{\alpha}_0$	$\dot{\gamma}_0$	x_0[m]	y_0
$\pi/3$	3	4	-3.660381	1.0	1.0

Table 3.6. Target and gain parameters in position control.

β_e $[rad]$	$\dot{\beta}_e$ $[rad/s]$	x_e[m]	y_e	k_3	k_4
$\pi/2$	0	0	0	10	5

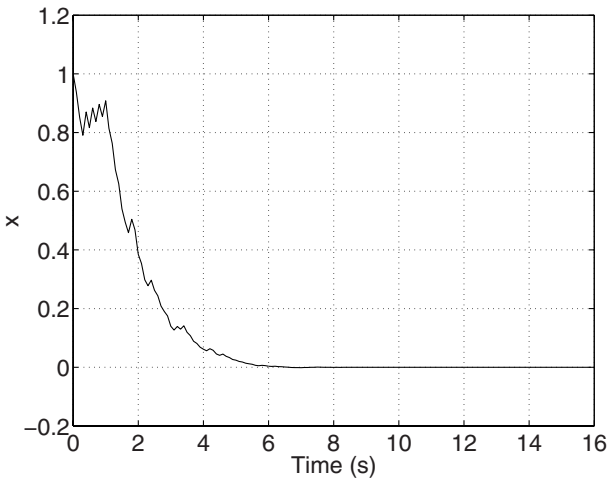


Fig. 3.24. Displacement in X.

Table 3.7. Initial parameters in line tracking.

β_0 $[rad]$	$\dot{\beta}_0$ $[rad/s]$	$\dot{\alpha}_0$	$\dot{\gamma}_0$	x_0[m]	y_0
$\pi/2.1$	0.1	4	-3.660381	0.05	0.05

Definition 4 $Uanh(.)$ is a unipolar function described as follows:

$$Uanh(x) = \frac{1}{1 + e^{-k_7 x}} \tag{3.59}$$

where k_7 is a positive scalar constant, which can be designed. $Uanh(.)$ is shown as in Figure 3.33.

We substitute $Sgn(.)$ and $\Theta(.)$ with $Tanh(.)$ and $Uanh(.)$, respectively. However, that they have different values is a problem, if $x = 0$. In a real time experiment, $x = 0$ very seldom appears and cannot be maintained, because there is too much noise. Thus, it is safe to perform the suggested substitution from this point of view.

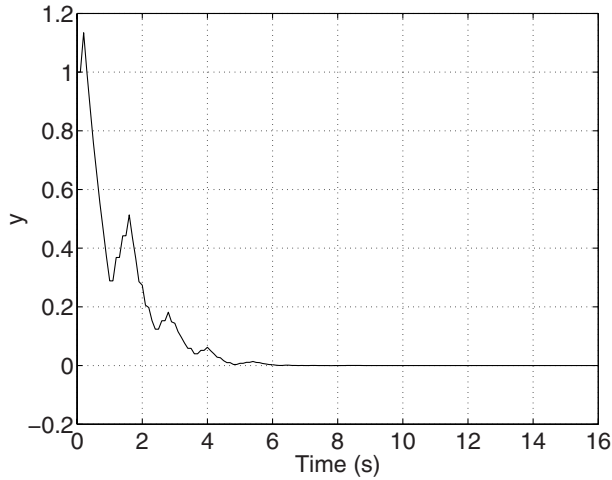


Fig. 3.25. Displacement in Y.

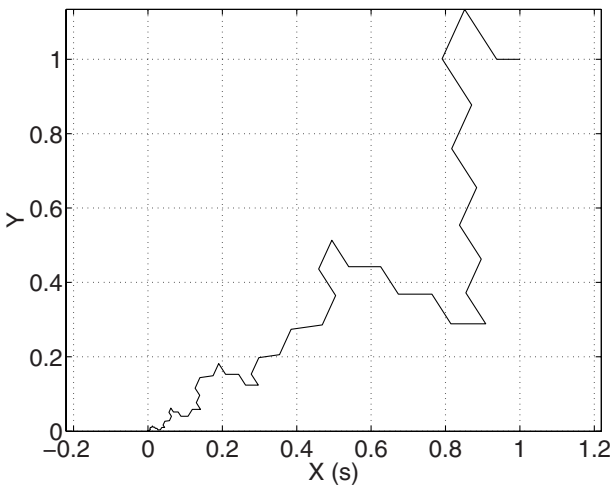


Fig. 3.26. X - Y of origin.

Table 3.8. Target and gain parameters in line tracking.

β_e $[rad]$	$\dot{\beta}_e$ $[rad/s]$	x_e[m]	y_e	k_3	k_5
$\pi/2$	0	3	4	10	5

3.3 Control Implementation

An on-board 100-MHZ 486 computer was installed in Gyrover to deal with on-board sensing and control. A flash PCMCIA card is used as the com-

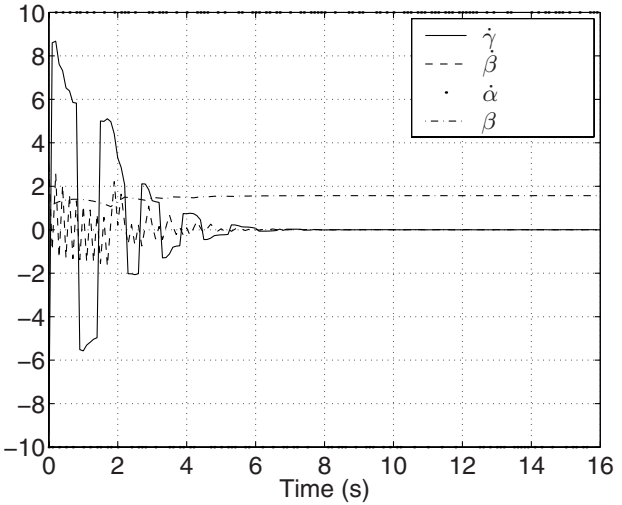


Fig. 3.27. The joint-space variables in position control.

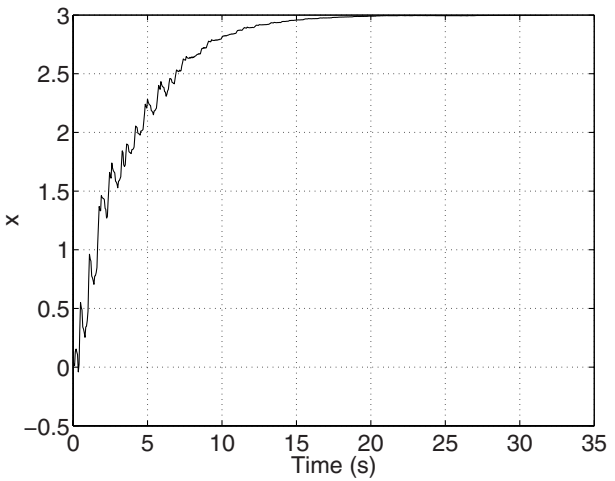


Fig. 3.28. Line tracking in X direction.

puter's hard disk and communicates with a stationary PC via a pair of wireless modems. Based on this communication system, we can download the sensor data file from the onboard computer, send supervising commands to Gyrover, and manually control Gyrover through the stationary PC. Moreover, a radio transmitter is installed for a human operator to remotely control Gyrover via the transmitter's two joysticks. One operator uses the transmitter to control the drive speed and tilt angle of Gyrover. Hence, we can record the operator's driving data.

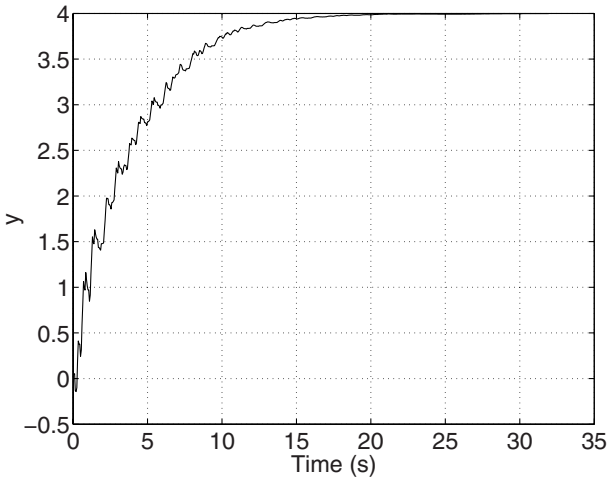


Fig. 3.29. Line tracking in Y direction.

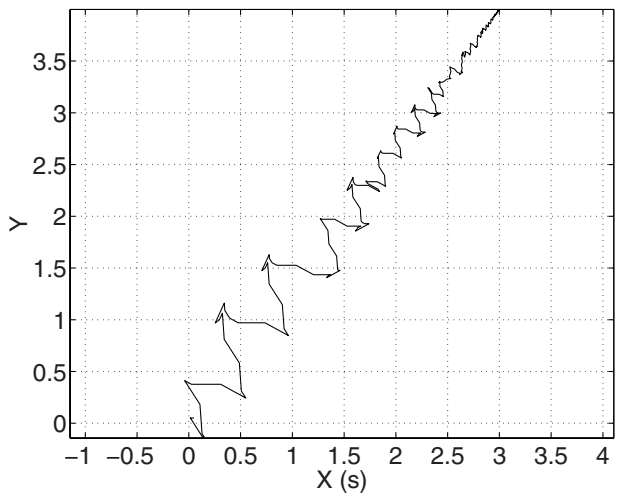


Fig. 3.30. X - Y of in line tracking.

Numerous sensors are installed in Gyrover to measure state variables (Figure 3.34). Two pulse encoders are installed to measure the spinning rate of the flywheel and the wheel. Furthermore, we have two gyros and an accelerometer to detect the angular velocity of yaw, pitch, roll, and acceleration respectively. A 2-axis tilt sensor has been developed and installed for directly measuring the lean angle and pitch angle of Gyrover. A gyro tilt potentiometer is used to calculate the tilt angle of the flywheel and its rate change.

The on-board computer runs on QNX, which is a real-time micro-kernel OS developed by QNX Software System Limited. Gyrover's software system

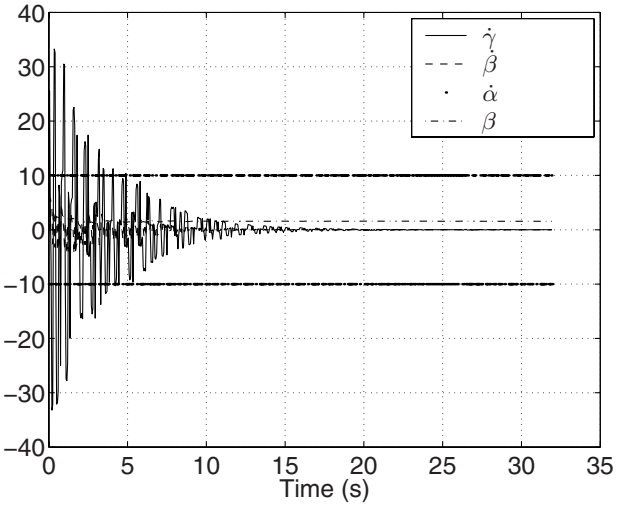


Fig. 3.31. The joint-space variables in line tracking.

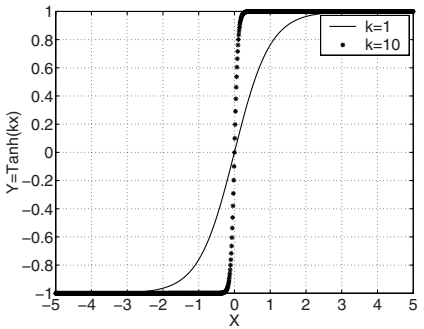


Fig. 3.32. Function Tanh(.).

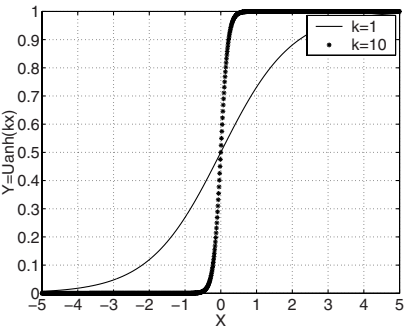


Fig. 3.33. Function Uanh(.).

is divided into three main programs: (1) communication server, (2) sensor server, and (3) controller. The communication server is used to communicate between the on-board computer and a stationary personal computer (PC) via an RS232, while the sensor server is used to handle all the sensors and actuators. The controller program implements the control algorithm and communicates among these servers. All programs run independently in order to allow real-time control of Gyrover. To compensate for the friction at the joints, we adopt the following approximate mathematical model:

$$F_f^{'} = \mu_v \dot{q} + [\mu_d + (\mu_s - \mu_d)\Gamma] sgn(\dot{q}) \tag{3.60}$$

where $\Gamma = diag\{e^{(-|\dot{q}_i|/D)}\}$, and

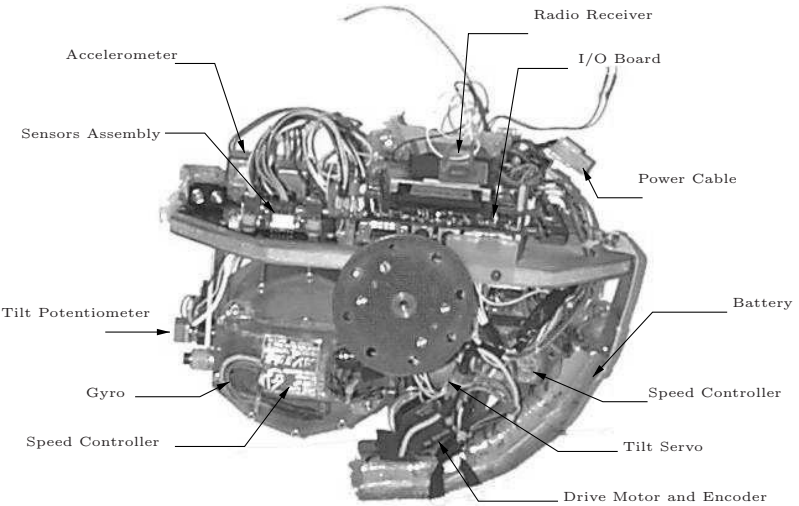


Fig. 3.34. Hardware configuration of Gyrover.

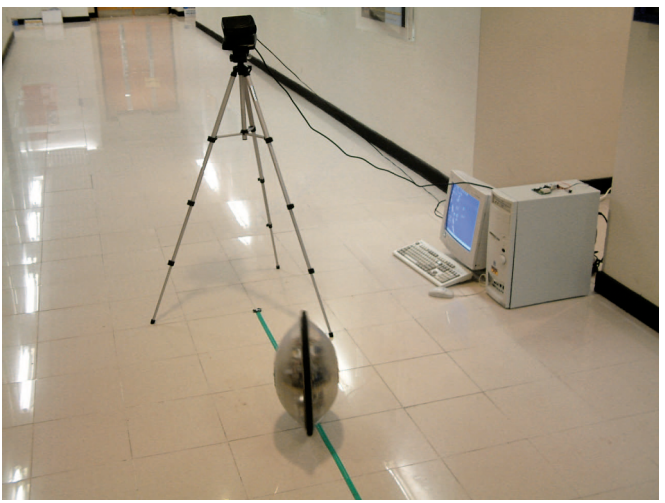


Fig. 3.35. Experiment in line following control.

$\mu_v \in R^{3\times3}$ is the viscous friction coefficient;
$\mu_d \in R^{3\times3}$ is the dynamic friction coefficient;
$\mu_s \in R^{3\times3}$ is the static friction coefficient;
We have $\mu_v = diag\{0.17, 0.15, 0.09\}$, $\mu_d = diag\{0.1, 0.1, 0.07\}$, $\mu_s = diag\{0.3, 0.25, 0.1\}$.

3.3.1 Vertical Balance

The purpose of this set of experiments is to keep Gyrover balanced. Some experimental results are shown in Figure 3.36 and Figure 3.37.

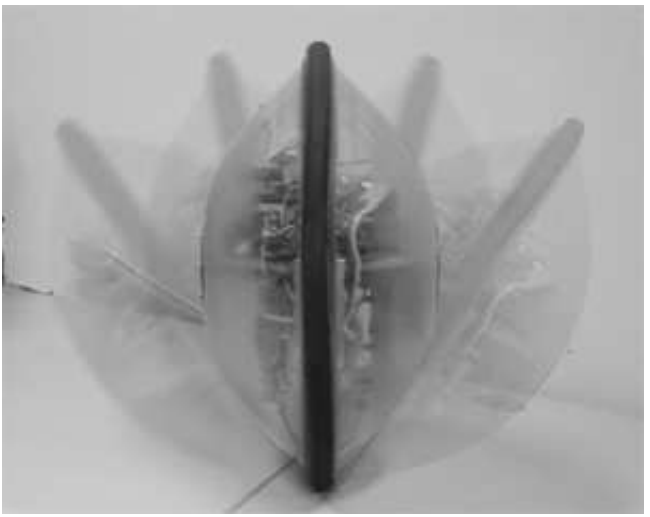


Fig. 3.36. Camera pictures in balance control.

Despite several successful examples, control laws sometimes may fail. We believe it is because the initial condition is set out of the domain D in Proposition 1. With regard to balance control, the initial condition seems to be too narrow for the constraint, $\sigma < \pi/2$. If we make a small modification to u_6 as follows, the controller can be used in a wider range of initial conditions.

$$u_6 = -((2+k_1)(\beta - \pi/2) + (3+2k_1)\dot{\beta} + (2+k_1)\ddot{\beta} + h_1(t)\dot{\beta} + h_2(t)u_5)/h_3(t)$$

where k_1 is a positive number, which can be designed. To prove the subsystem $\beta, \dot{\beta}, \ddot{\beta}$ is asymptotically stabilized, we consider the following Lyapunov function candidate

$$V^* = (\beta - \pi/2)^2/2 + (\dot{\beta} + \beta - \pi/2)^2/2 + (\ddot{\beta} + (1+k_1)(\dot{\beta} + \beta - \pi/2))^2/2.$$

Its derivative is

$$\dot{V}^* = -(\beta - \pi/2)^2 - k_1(\dot{\beta} + \beta - \pi/2)^2 - (\ddot{\beta} + (1+k_1)(\dot{\beta} + \beta - \pi/2))^2.$$

3.3.2 Position Control

In this experiment, Gyrover is required to move from a Cartesian space point (x_0, y_0) to the original point of Cartesian space, where $x_0 = 3$ m and $y_0 = 4$ m.

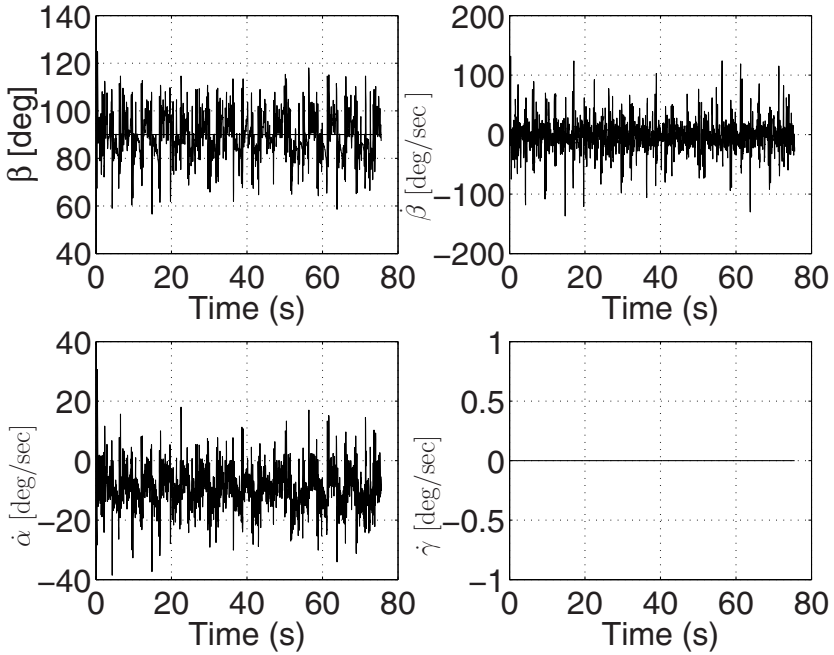


Fig. 3.37. Sensor data in balance control.

We mount a high resolution 2/3" ccd, format CANON COMMUNICATION CAMERA VC-C1 on a tripod. The camera has been calibrated and we have mapped the field of vision to the Cartesian space coordinate which is anchored on the ground. The camera has a pixel array of $768(H) \times 576(V)$. In order to communicate the data, an interface board (digital I/O) is installed in a PC and there are wireless modems to connect the PC and Gyrover.

The first problem experienced is that the system states exhibited highly oscillatory behavior and the second is that control inputs sometimes needed to switch too sharply and fast. Both will cause difficulties for real time control and result in worse performance. To solve these problems, we propose to replace the sign functions in the controllers with tanh functions.

Let $Tanh(.)$ be a bipolar function described as follows:

$$Tanh(x) = \frac{1 - e^{-k_6 x}}{1 + e^{-k_6 x}} \tag{3.61}$$

where k_6 is a positive scalar constant, which can be designed.

Let $Uanh(.)$ be a unipolar function described as follows:

$$Uanh(x) = \frac{1}{1 + e^{-k_7 x}} \tag{3.62}$$

where k_7 is a positive scalar constant, which can be designed.

We substitute $Sgn(.)$ and $\Theta(.)$ with $Tanh(.)$ and $Uanh(.)$, respectively. However, that they have different values is a problem, if $x = 0$. In a real time experiment, $x = 0$ very seldom appears and cannot be maintained, because there is too much noise. Thus, it is safe to perform the suggested substitution from this point of view.

The trajectory that Gyrover traveled is shown in Figure 3.38. A number of sensor reading results are shown in Figure 3.39.

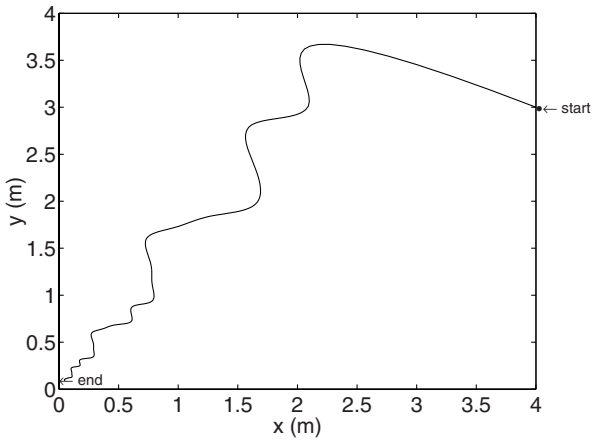


Fig. 3.38. Trajectories in point-to-point control.

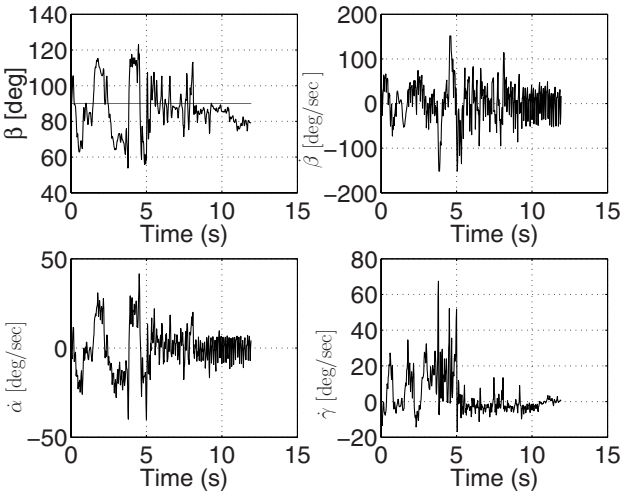


Fig. 3.39. Sensor data in point-to-point control.

3.3.3 Path Following

In this experiment, Gyrover is required to travel a straight path which is about 5 m long. The trajectory that Gyrover traveled is shown in Figure 6.11. A number of sensor reading results are shown in Figure 6.16.

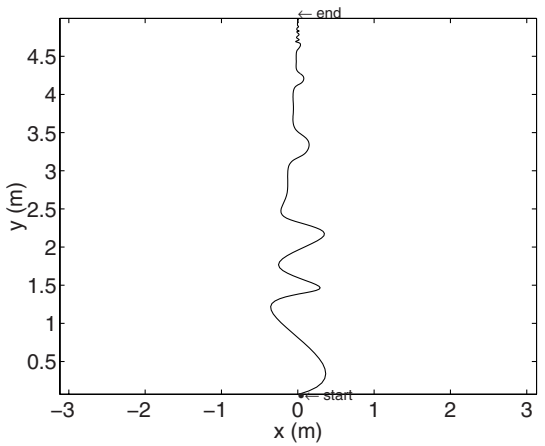


Fig. 3.40. Trajectories in the straight path test.

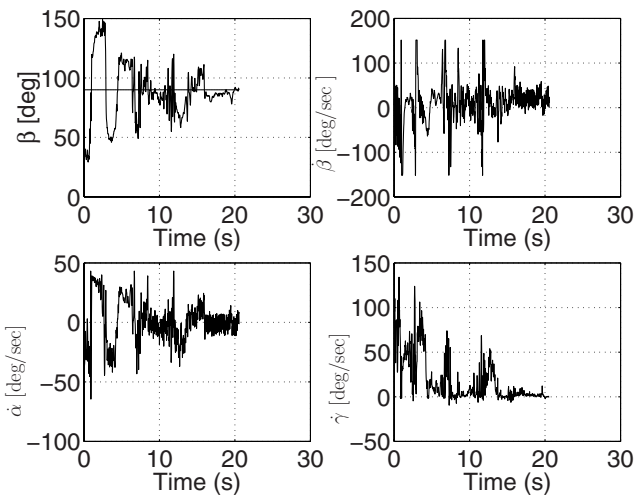


Fig. 3.41. Sensor data in the straight path test.

4

Learning-based Control

4.1 Learning by CNN

Due to the complexity of the system, it is difficult for us to work out a "complete" analytical model of it. Therefore, in this chapter, we propose using a machine learning algorithm, Cascade Neural Network (CNN) with node-decoupled extended Kalman Filtering (NDEKF), to model the robot's behaviors from human control strategy (HAS).

Motivation

Gyrover is a single track mobile vehicle which is inherently unstable in the lateral direction. With the lack of a wide polygon of support (single-point contact with the ground), Gyrover has very bad static stability, even though it is equipped with an internal gyroscope spinning at a high rate. The thin pneumatic tire which is wrapped around the robot makes it difficult to stand in a stationary position for a very long time, it will fall on the ground. However, by tilting the internal gyroscope into different orientations, we can indirectly control the lean angle of the robot, which implies that it is possible for us to keep the robot to stay in its upright position with a proper control method.

Previous research of Gyrover has been focused on dynamics and control, including the kinematic constraints and motion equations [74, 107, 115, 6, 5, 113, 114]. However, the robot concept brings a number of challenging problems in modeling and control because of the highly coupled dynamics, the nonholonomic constraints and the non-minimum phase behavior of the system. The proposed linear state feedback model in [6] only guarantees the local stability of the system. Moreover, the dynamic model derived has been based on many assumptions which may not be realistic.

In [5], a linear state feedback controller is developed for stabilizing the robot to any desired angle, however, this model only applied for the case when the robot reaches steady state. By consideration of the swinging motion of the

Y. Xu and Y. Ou: Control of Single Wheel Robots, STAR 20, pp. 73–117, 2005.

internal mechanism, the model is modified in [113]. Unfortunately, the models obtained above are based on the assumption of rolling without slipping, that is, the robot must be rolling perfectly on the ground. Therefore, these models are not applicable for the static situation. In the static situation, the coupling between the wheel and the flywheel becomes much more complicated, which makes it difficult to derive an analytical model by traditional control methods.

On the other hand, humans are capable of mastering complex and highly nonlinear control systems. A typical example is car driving. For Gyrover control, humans are able to control the robot well if enough practice is undertaken. Thus, we intuitively come up with the idea of machine learning, a model-free approach to model this kind of human control strategy. This approach is suitable for Gyrover control for the following reasons:

- Gyrover is a complex system, for which it is difficult for us to develop a complete dynamic model to represent the robot's behaviors by using traditional control methods.
- From a practical point of view, it is equally difficult to model the system precisely due to some unmodeled factors, such as friction. Friction is an important issue when we are dealing with the coupling between the wheel and the spinning flywheel.
- Although Gyrover is a complex system, humans can control the robot through a radio transmitter to perform various kinds of task. They do not need to explicitly model a system in order to control it. Through interaction with the system and observation of the behaviors of the system, humans are able to "learn" how to control a system.
- The learning process is in fact a direct input-output mapping between the system sensory data and the actuation data. A controller is generated by using the training data while a human "teacher" controls the system until the synthesized controller can perform the same way as a human.

4.1.1 Cascade Neural Network with Kalman Filtering

The field of intelligent control has emerged from the field of classical control theory to deal with applications which are too complex for classical control approaches. In terms of complexity, human control strategy lies between low-level feedback control and high-level reasoning, and encompasses a wide range of useful physical tasks with a reasonably well-defined numeric input/output representation.

Here, we introduce a continuous learning architecture for modeling human control strategies based on a neural network. Since most neural networks used today rely on rigid, fixed architecture networks and/or with slow gradient descent-based training algorithms, they may not be a suitable method to model the complex, dynamic and nonlinear human control strategy. To counter these problems, a new neural network learning architecture is proposed in [78], which combines (1) flexible cascade neural networks, which

dynamically adjust the size of the neural network as part of the learning process, and (2) node-decoupled extended Kalman Filtering (NDEKF), a faster converging alternative to exappropriation. This methodology has been proved which can efficiently model human control skills [76], [116] and human sensation [62].

First of all, let's discuss the architecture of cascade learning. In cascade learning, the network topology is not fixed prior to learning, hidden units are added to an initially minimal network one at a time. This not only frees us from a prior choice of network architecture, but also allows new hidden units to assume variable activation functions. That is, each hidden unit's activation function no longer needs to be confined to just a sigmoidal nonlinearity. A priori assumption about the underlying functional form of the mapping we wish to learn is thus minimized. The whole training process is described below:

1. Initially, no hidden unit exists in the network, only direct input-output connections. These weights are trained first, to obtain a linear relationship, if any.
2. With no further significant decrease in the RMS error (e_{RMS}), a first hidden node will be introduced into the network from a pool of candidate units. These candidate units are trained independently and in parallel with different random initial weights by using the quickprop algorithm.
3. The best candidate unit will be selected and installed into the network if no more appreciable error reduction occurs, therefore, the first hidden node is produced.
4. Once the hidden unit is installed, all the input weights of the hidden unit will be frozen, while the weights to the output unit(s) is/are going to train again. This allows for a much faster convergence of the weights during training than a standard multi-layer feedforward network.
5. This process (from step 2 - step 4) is repeated until the e_{RMS} reduces sufficiently for the training set or the number of hidden units reaches a predefined maximum number.

Figure 4.1 illustrates, for example, how a two-input, single-output network with a bias unit grows with increasing number of hidden nodes.

A cascade neural network with n_{in} input units (including the bias unit), n_h hidden units, and n_{out}, has n_w connections (total number of weights) where,

$$n_w = n_{in}n_{out} + n_h(n_{in} + n_{out}) + (n_h - 1)n_h/2 \tag{4.1}$$

In fact, any multi-layer feedforward neural network with k hidden units arranged in m layers, fully connected between consecutive layers, is a special case of a cascade network with k hidden units with some of the weights equal to zero. Thus, this architecture relaxes a prior assumptions about the functional form of the model to be learnt by dynamically adjusting the network size. We can further relax these assumptions by allowing new hidden units to have different activation functions. The kind of activation functions which

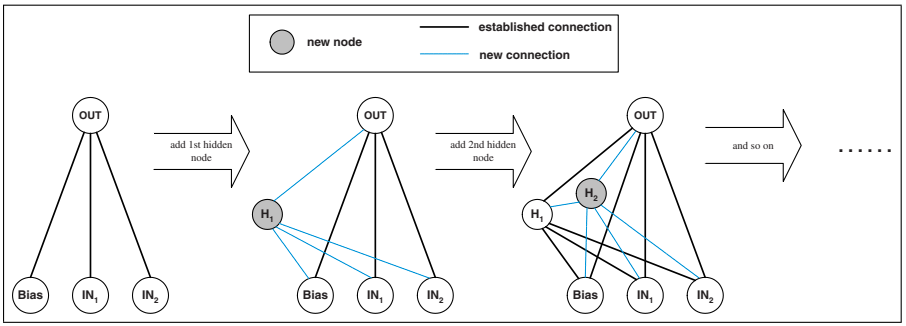


Fig. 4.1. The cascade learning architecture.

reduces e_{RMS} most will be selected during the process, Sigmoid, Gaussian, and sinusoidal functions of various frequency are some of the available types of activation functions we can choose.

While quickprop is an improvement over the standard exappropriation algorithm for adjusting the weights in the cascade network, it still requires many iterations until satisfactory convergence is reached. When combining cascade neural networks with node-decoupled extended Kalman filtering (NDEKF), [76] has shown that this methodology can solve the poor local minima problem, and that the resulting learning architecture substantially outperforms other neural network training paradigms in learning speed and/or error convergence for learning tasks important in control problems.

4.1.2 Learning architecture

Denote ω_k^i as the input-side weight vector of length n_w^i at iteration k, for $i \in \{0, 1, \ldots, n_o\}$, and,

$$n_w^i = \begin{cases} n_{in} + n_h - 1 & i = 0 \\ n_{in} + n_h & i \in \{1, \ldots, n_o\} \end{cases} \tag{4.2}$$

The NDEKF weight-update recursion is given by, (starting from equation (3.6) to (3.9), {}'s, ()'s and []'s evaluate to scalars, vectors and matrices respectively)

$$\omega_{k+1}^i = \omega_k^i + \{(\psi_k^i)^T (A_k \xi_k)\} \phi_k^i \tag{4.3}$$

where ξ_k is the n_o-dimensional error vector for the current training pattern, ψ_k^i is the n_o-dimensional vector of partial derivatives of the network's output unit signals with respect to the ith unit's net input, and,

$$\phi_k^i = P_k^i \zeta_k^i \tag{4.4}$$

$$A_k = \left[I + \sum_{i=0}^{n_o} \{(\zeta_k^i)^T \phi_k^i\}[\psi_k^i (\psi_k^i)^T] \right]^{-1} \tag{4.5}$$

$$P_{k+1}^i = P_k^i - \{(\psi_k^i)^T (A_k \psi_k^i)\}[\phi_k^i (\phi_k^i)^T] + \eta_Q I \tag{4.6}$$

$$P_0^i = (1/\eta_P) I \tag{4.7}$$

where ζ_k^i is the n_w^i-dimensional input vector for the ith unit, and P_k^i is the $n_w^i \times n_w^i$ approximate conditional error covariance matrix for the ith unit. The parameter η_Q is introduced in (3.9) to avoid the singularity problems for error covariance matrices. Throughout the training, we use $\eta_Q = 0.0001$ and $\eta_P = 0.01$.

The vector ψ_k^i can be computed in this way: let O_i be the value of the ith output node, Γ_O be its corresponding activation function, net_{Oi} be its net activation, Γ_H be the activation function for the current hidden unit being trained, and net_H be its net activation. We have,

$$\frac{\partial O_i}{\partial net_{Oj}} = 0, \forall i \neq j \tag{4.8}$$

$$\frac{\partial O_i}{\partial net_{Oi}} = \Gamma'_O(net_{Oi}), i \in \{1, \ldots, n_o\} \tag{4.9}$$

$$\frac{\partial O_i}{\partial net_H} = w_{Hi} \cdot \Gamma'_O(net_{Oi}) \cdot \Gamma'_H(net_H) \tag{4.10}$$

where w_{Hi} is the weight connecting the current hidden node to the ith output node.

4.1.3 Model evaluation

The main advantage of modeling a robot's behaviors by learning, is that no explicit physical model is required, however, this also presents its biggest weakness. Since a model is trained by the input-output relationship only, the lack of a scientific justification degrades the confidence that we can show in these learnt models. This is especially true when the process we are going to model is dynamic and stochastic in nature, which is the case in human control strategy. For a dynamic process, errors may feed back into the model to produce outputs which are not characteristics of the original process or make the process unstable. For a stochastic process, a static error criterion such as RMS error, based on the difference between the training data and the predicted model outputs is inadequate to gauge the fidelity of a learnt model to the source process.

In general, for different models, the similarity between a dynamic human control trajectory and a model-generated one will vary continually, from completely dissimilar to nearly identical. Furthermore, one cannot expect exact trajectories for the system and the learnt model, even if equivalent initial

conditions are given. To effectively evaluate the learnt models, we introduce a stochastic similarity measure proposed in [77]. This method is based on Hidden Markov Model (HMM) analysis, which is a useful tool for comparing stochastic, dynamic and multi-dimensional trajectories.

Hidden Markov Model is a trainable statistical model, which consists of a set of n states, interconnected by probabilistic transitions, each of these states has some output probability distributions associated with it. A discrete HMM is completely defined by,

$$\lambda = \{A, B, \pi\} \tag{4.11}$$

where A is the probabilistic $n_s \times n_s$ state transition matrix, B is the $L \times n_s$ output probability matrix with L discrete output symbols $l \in \{1, 2, \ldots, L\}$, and π is the n-length initial state probability distribution vector for HMM. Two HMMs (λ_1 and λ_2) are said to be equivalent if and only if,

$$P(O|\lambda_1) = P(O|\lambda_2), \forall O \tag{4.12}$$

We prefer discrete HMMs rather than continuous or semi-continuous HMMs, because they are relatively simple in computation and less sensitive to initial random parameter settings. However, the human control trajectories we are going to measure are continuous and real-valued functions. In order to make use of the discrete HMMs, we must convert the data sets into sequences of discrete symbols O_n by the following procedures:

1. Normalization
2. Spectral conversion
3. Power Spectral Density (PSD) estimation
4. Vector quantization

The purpose of steps (1) - (3) is to extract some meaningful feature vectors V for the vector quantizer. In step (4), the feature vectors V are converted to L discrete symbols, where L is the number of output observable in our HMMs.

In general, assume that we are going to compare the observation sequences ($\bar{O}_1$ and $\bar{O}_2$) from two stochastic processes (Γ_1 and Γ_2). The probability of the observation sequences $\bar{O}_i$ given the HMM model λ_j, is given by [77],

$$P_{ij} = P(\bar{O}_i|\lambda_j)^{1/\bar{T}_i}, \qquad i, j \in \{1, 2\} \tag{4.13}$$

where the above equation is normalized with respect to the total numbers of symbols $\bar{T}_i$.

The similarity measure σ between $\bar{O}_1$ and $\bar{O}_2$ is,

$$\sigma(\bar{O}_1, \bar{O}_2) = \sqrt{\frac{P_{12}P_{21}}{P_{11}P_{22}}} \tag{4.14}$$

Figure 4.2 illustrates the overall approach to evaluate the similarity between two observation sequences. The HMMs are trained by each observation sequence first, then we cross-evaluate each observation sequence on the other

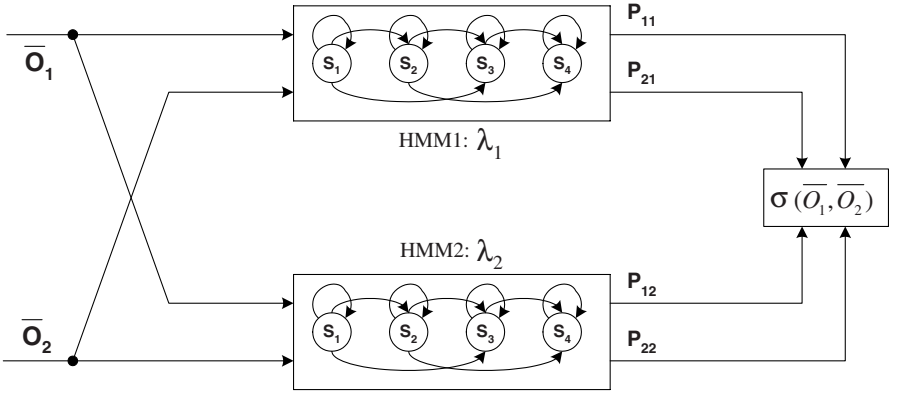


Fig. 4.2. Similarity measure between $\bar{O}_1$ and $\bar{O}_2$.

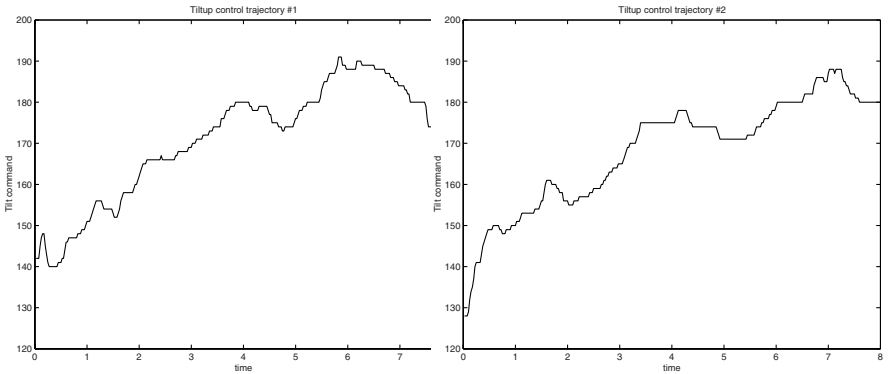
HMM. Based on the four normalized probabilities, the similarity measure σ can be obtained.

Here, we demonstrate an example of how this similarity measure works. Figure 4.3 shows four Gyrover control trajectories. Figure 4.3(a) and 4.3(b) correspond to the tiltup motion control, while Figure 4.3(c) and 4.3(d) correspond to the lateral stabilization control of Gyrover. We applied the HMM similarity measure across these four trajectories. We might expect that the trajectories of the same motion should have a relatively high similarity and that any two trajectories which were generated from different kinds of motion should have a low similarity value. We summarize the results in Table 4.1.

Table 4.1. Similarity measures between different control trajectories.

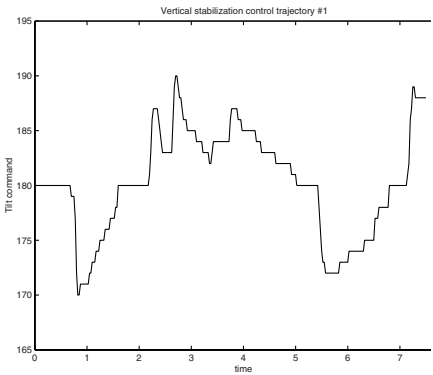
	Tiltup #1	Tiltup #2	Vertical stab. #1	Vertical stab. #2
Tiltup #1	1.000	0.6306	0.0524	0.1234
Tiltup #2	0.6306	1.000	0.0615	0.0380
Vertical stab. #1	0.0524	0.0615	1.000	0.4994
Vertical stab. #2	0.1234	0.0380	0.4994	1.000

From Table 4.1, it is clear that this similarity measure can accurately classify dynamic control trajectories from the same type of motion, while discriminating those from different motions by giving a low similarity value. This similarity measure can be applied towards validating a learned model's fidelity to its training data, by comparing the model's dynamic trajectories in the feedback loop to the human's dynamic control trajectories.

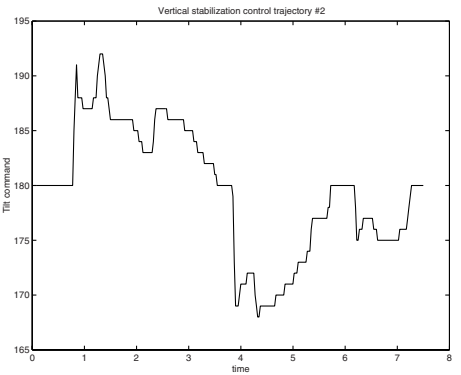


(a) Tiltup 1

(b) Tiltup 2



(c) Vertical balancing 1



(d) Vertical balancing 2

Fig. 4.3. Control data for different motions.

4.1.4 Training procedures

Fist of all, we have made two assumptions for the training data provided for the learning process:

1. **Reliable training set.** Since learning is a kind of high-level, model free "teaching by showing" approach, the stability or robustness of the learnt model heavily depends on the operating skills of a "human teacher", in order to provide reliable and stable control. Therefore, throughout the teaching process, we assume that the operator is skillful and experienced enough to master the robot. That is, the training data can fully reflect the skills in a particular robot's behavior. Besides the quality of the training data, the quantity of the data points is equally important. If the training set is in a larger scale, a more complete skill can be described.

2. **Injective mapping.** Another important issue is about the mappings between inputs and outputs in a static map. Figure 4.4 shows a human control strategy for the lateral balancing behavior, it is not difficult to figure out that the control of the flywheel is always switching (a very sharp change). That is, at a short moment ago, the command is positive, but in the next moment, the command will change to negative. Unfortunately, the switching problem causes very similar inputs to be mapped to radically different outputs, which is difficult for the cascade neural network to adapt to, Figure 4.5. To ensure that there will be a correct mapping, enough time-delayed histories should be provided in the training data set. In our cascade network training, we will provide at least 20 pieces of history data ($n_i \geq 20$) to guarantee the infectiveness of the mapping.

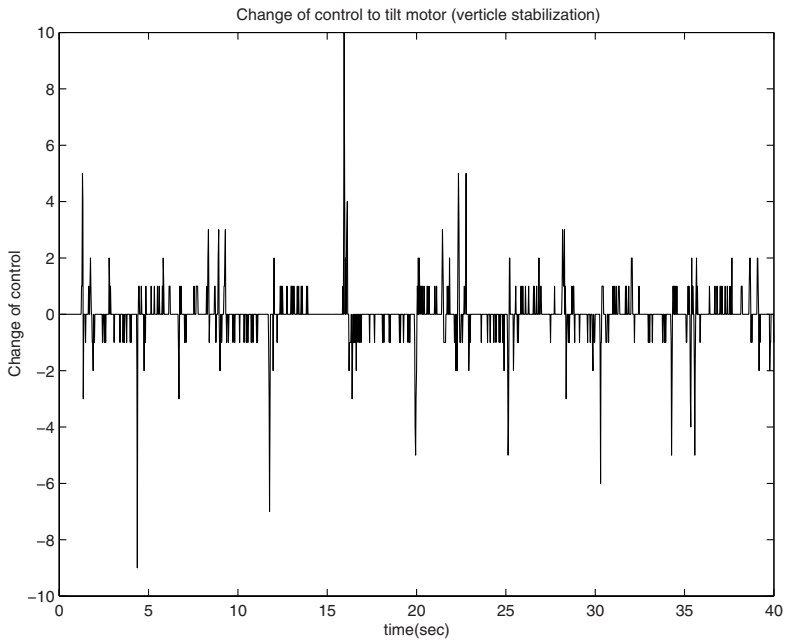


Fig. 4.4. Switchings in human control of the flywheel.

For each model, we process the training data as follows:

1. **Removal of irrelevant data**
   Let $[t, t+t_m]$ denote an interval of time, in seconds, that a human operator has given an inappropriate command during the experiment. Then, we cut the data corresponding to the time interval $[t-1, t+t_m]$ from the training data. In other words, we not only remove the irrelevant data from the training set, but also the second data leading to the inappropriate

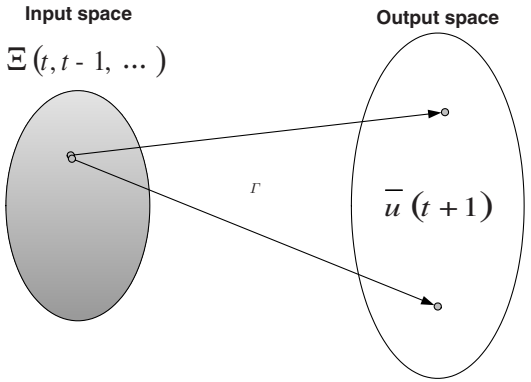


Fig. 4.5. Similar inputs can be mapped to extreme different outputs if switching occurs.

command time interval. This ensures that the cascade model does not learn control behaviors that are potentially destabilizing.

2. **Normalization**
   We normalize each input dimension of the training data, such that all the input in the training data falls inside the interval $[-1, 1]$.
3. **Generate time-shifted data**
   As mentioned in the previous section, we need to provide enough time-delayed values of each state and control variable such that the model is able to build necessary derivative dependencies between the inputs and outputs. In our cascade network training, we will provide 20 pieces of history data.
4. **Randomization**
   Finally, we randomize the input-output training vectors and select half for training, while reserving the other half for testing.

The sampling rate of the training data is 40Hz and a typical training set will consist of approximately 10,000 data points.

4.2 Learning by SVM

4.2.1 Support Vector Machines

Support Vector Machine

Originally, SVM was developed from classification problems. It was then, extended to regression estimation problems, i.e., to problems related to find the function $y = f(\mathbf{x})$ given by its measurements y_i with noise at some (usually random) vector $\mathbf{x}$

$$(y_1, \mathbf{x}_1), ..., (y_l, \mathbf{x}_l). \tag{4.15}$$

where each of them is generated from an unknown probability distribution $P(\mathbf{x}, y)$ containing the underlying dependency. (Here and below, bold face characters denote vectors.)

In this paper, the term SVM will refer to both classification and regression methods, and the terms Support Vector Classification (SVC) and Support Vector Regression (SVR) will be used for specification. In SVR, the basic idea is to map the data X into a high-dimensional feature space $\mathcal{F}$ via a nonlinear mapping Φ, and to do linear regression in this space [108].

$$f(\mathbf{x}) = (\omega \cdot \Phi(\mathbf{x})) + b \quad \text{with } \Phi : R^n \rightarrow \mathcal{F},\ \omega \in \mathcal{F}, \tag{4.16}$$

where b is a threshold. Thus, linear regression in a high dimensional (feature) space corresponds to nonlinear regression in the low dimensional input space R^n. Note that the dot product in Equation (4.16) between ω and $\Phi(\mathbf{x})$ would have to be computed in this high dimensional space (which is usually intractable). If we are not able to use the kernel that eventually leaves us with dot products that can be implicitly expressed in the low dimensional input space R^n. Since Φ is fixed, we determine ω from the data by minimizing the sum of the empirical risk $R_{emp}[f]$ and a complexity term $\parallel \omega \parallel^2$, which enforces flatness in feature space

$$R_{reg}[f] = R_{emp}[f] + \lambda \parallel \omega \parallel^2 = \sum_{i=1}^{l} C(f(\mathbf{x}_i) - y_i) + \lambda \parallel \omega \parallel^2, \tag{4.17}$$

where l denotes the sample size $(\mathbf{x}_1, ..., x_l)$, $C(.)$ is a loss function and λ is a regularization constant. For a large set of loss functions, Equation (4.17) can be minimized by solving a quadratic programming problem, which is uniquely solvable [98]. It can be shown that the vector ω can be written in terms of the data points

$$\omega = \sum_{i=1}^{l} (\alpha_i - \alpha^*) \Phi(\mathbf{x}_i), \tag{4.18}$$

with α_i, α_i^* being the solution of the aforementioned quadratic programming problem [108]. The positive Lagrange multipliers α_i and α_i^* are called support values. They have an intuitive interpretation as forces pushing and pulling the estimate $f(\mathbf{x}_i)$ towards the measurements y_i [27]. Taking Equation (4.18) and Equation (4.16) into account, we are able to rewrite the whole problem in terms of dot products in the low dimensional input space

$$\begin{aligned} f(\mathbf{x}) &= \textstyle\sum_{l=1}^{l} (\alpha_i - \alpha_i^*)(\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})) + b \\ &= \textstyle\sum_{l=1}^{l} (\alpha_i - \alpha_i^*) k(\mathbf{x}_i, \mathbf{x}) + b. \end{aligned} \tag{4.19}$$

In Equation (4.19), we introduce a kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. As explained in [15], any symmetric kernel function k satisfying Mercer's condition corresponds to a dot product in some feature space.

For a more detailed reference on the theory and computation of SVM, readers are referred to [27].

4.2.2 Learning Approach

The skill that we are considering here is the control strategy demonstrated by a human expert to obtain a certain control target. For example, in controlling a robotic system, a human expert gives commands by way of a controller, such as a joystick, and the robot executes the task. The desired trajectory of the robot given by an expert through a joystick reflects the expert's control strategy. The goal of the human control strategy learning here is to model the expert control strategy and according to current states select one command that represents the most likely human expert strategy. Here, we consider the human expert actions as the measurable stochastic process and the strategy behind it as the mapping between the current states and commands. An SVR is employed to represent human expert strategy for the given task, and the model parameters are sought through an off-line learning process. This method allows human experts to transfer their control strategy to robots. The procedure for SVM-based control strategy learning can be summarized as follows:

1. Representing the control strategy by an SVM: Choosing a suitable kernel and structure of an SVM for characterizing the control strategy.
2. Collecting the training data: Obtaining the data representing the control strategy we want to model.
3. Training the model: Encoding the control strategy into an SVM.
4. Finding the best human performance: Learning/transferring the control strategy.

For training an SVM learning controller, the system states will usually be treated as the learning inputs and the control inputs/commands will be the learning outputs.

Training Example Collection

An SVM does not require a large amount of training samples as do most ANNs. Scholkopf [23] pointed out that the *actual risk* $R(w)$ of the learning machine is expressed as:

$$R(w) = \int \frac{1}{2} \| f_w(\mathbf{x}) - y \| dP(\mathbf{x}, y). \tag{4.20}$$

where $P(\mathbf{x}, y)$ is the same distribution defined in (4.15). The problem is that $R(w)$ is unknown, since P is unknown.

The straightforward approach to minimize the *empirical risk*,

$$R_{emp}(w) = \frac{1}{l} \sum_{i=1}^{l} |f_w(\mathbf{x}_i) - y_i\|,$$

turns out not to guarantee a small actual risk $R(w)$, if the number l of training examples is limited. In other words: a small error in the training set does

not necessarily imply a high *generalization* ability (i.e., a small error on an independent test set). This phenomenon is often referred to as *overfitting*. To solve the problem, novel statistical techniques have been developed during the last 30 years. For the learning problem, the *Structure Risk Minimization* (SRM) principle is based on the fact that for any $w \in \Lambda$ and $l > h$, with a probability of at least 1-η, the bound

$$R(w) \leq R_{emp}(w) + \Phi(\frac{h}{l}, \frac{log(\eta)}{l}) \tag{4.21}$$

holds, where the *confidence term* Φ is defined as

$$\Phi(\frac{h}{l}, \frac{log(\eta)}{l}) = \sqrt{\frac{h(log\frac{2l}{h} + 1) - log(\eta/4)}{l}}.$$

The parameter h is called the VC (*Vapnik-Chervonenkis*) dimension of a set of functions, which describes the capacity of a set of functions. Usually, to decrease the $R_{emp}(w)$ to some bound, most ANNs with complex mathematical structure have a very high value of h. It is noted that when n/h is small (for example less than 20, the training sample is small in size), Φ has a large value. When this occurs, performance poorly represents $R(w)$ with $R_{emp}(w)$. As a result, according to the SRM principle, a large training sample size is required to acquire a satisfactory learning machine. However, by mapping the inputs into a feature space using a relatively simple mathematical function, such as a polynomial kernel, an SVM has a small value for h, and at the same time maintains the $R_{emp}(w)$ in the same bound. To decrease $R(w)$ with Φ, therefore, requires small n, which is enough also for small h.

Training process

The first problem, before beginning training, is to choose between SVR or SVC; the choice being dependent on special control systems. For example, our experimental system Gyrover has the control commands U_0 and U_1. Their values are scaled to the tilt angle of the flywheel and the driving speed of Gyrover respectively. Thus, for this case, we will choose SVR. However, for a smart wheelchair system, for instance, the control commands 1, 2, 3 and 4 correspond to *"go ahead", "turn left", "turn right", and "stop"* respectively. Since this is a classification problem, it is better to choose SVC.

The second problem is to select a kernel. There are several kinds of kernel appropriate for an SVM [27]. The main function of kernels in an SVM is to map the input space to a high dimensional feature space, and at that space the feature elements have a linear relation with the learning output. In most cases, it is difficult to set up this kind of linear relations. This is because usually we do not know the mathematical relation between the input and output. Thus, it is better to test more kernels and choose the best one.

The third problem is about "scaling". Here, scaling refers to putting each Column's data into a range between -1 and 1. It is important to void the outputs if they are seriously affected by some states, for the "unit's" sake. Moreover, if some states are more important than others, we may enlarge their range to emphasize their effect.

Time Consideration

An SVM usually requires more time in obtaining the optimal weight matrix for the same set of training samples than a general neural network learner, such as Exappropriation neural networks (BPNN). An SVM determines the weight matrix through an optimization seeking process, whereas an ANN determines it by modifying the weights matrix backward from the differences between the actual outputs and estimated outputs. [15] and Vapnik [108] show how training an SVM for the pattern recognition problem (for a regression problem it is similar but a little more complex) leads to the following quadratic optimization problem (QP)OP1.

$$(\mathbf{OP1}) \text{ minimize}: \quad W(\alpha) = -\sum_{i=1}^{l} \alpha_i + \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{4.22}$$

$$\text{subject to}: \quad \sum_{i=1}^{l} y_i \alpha_i = 0 \quad \forall i \ : \ 0 \leq \alpha_i \leq C \tag{4.23}$$

The number of training examples is denoted by l. α is a vector of l variables, where each component α_i corresponds to a training example $(\mathbf{x}_i, y_i)$. C is the boundary of α_i. The solution of OP1 is the vector $\hat{\alpha}$ for which (4.22) is minimized and the constraints (4.23) are fulfilled. Defining the matrix $\tilde{Q}$ as $(\tilde{Q})_{ij} = y_i y_j k(x_i, x_j)$, this can be equivalently written as

$$\text{minimize}: \quad W(\alpha) = -\alpha^T \mathbf{1} + \frac{1}{2} \alpha^T \tilde{\mathbf{Q}} \alpha \tag{4.24}$$

$$\text{subject to}: \quad \alpha^T \mathbf{y} = 0 \quad , \ 0 \leq \alpha \leq C\mathbf{1} \tag{4.25}$$

Since the size of the matrix $\tilde{Q}$ is l^2, the size of the optimization problem depends on the number of training examples. However, most general ANNs, (such as BPNN) have linear relationship with the sample size. Because of this, SVMs usually need more training time than most other ANNs. Moreover, if the size of the sample is large, this phenomenon is much more serious. For example, for a size of 400 example set, an SVM needs about 10 minutes to complete the training process and a BPNN requires about 1 minute will finish. If the size of the sample reaches $10,000$, an SVM will need more than 100 hours to complete, whereas, the BPNN just requires about 25 minutes.

Learning Precise Consideration

In practice applications, the control process of an SVM learner is much smoother than general ANN learners. The reason for this is that many ANNs and HMM methods exhibit the local minima problem, whereas for an SVM the quadratic property guarantees the optimization functions are convex. Thus it must be the global minima.

Moreover, for the class of dynamically stable, statically unstable robots such as Gyrover, high learning precision is required and very important. This is because the learning controller will be returned to control the robot and form a new dynamic system. Larger errors in control inputs will cause system instability and control process failure. During our experiments, we always required several training sessions to produce a successful general ANN controller. However, after training the SVM learning controller it always worked well immediately.

4.2.3 Convergence Analysis

Here, we provide the convergence analysis of the learning controller, but not the learning convergence of SVMs. From the practice point of view, we consider the local asymptotical convergence instead of global or semi-global convergence. This is because, in the training process, we can only collect data in a region of systems variables, called the "working space". It is a set of all the points that the system states can reach. Furthermore, we define "region of operation" $\mathcal{D}$ as a meaningful working region, a subset of "working space". Usually, region of operation $\mathcal{D}$ is a compact subset of the full space R^n with its origin. Thus, with these local data, we almost can not obtain a highly precise training result in the full space R^n.

In fact, for this class of learning controller, it can only drive the system to converge into a bounded small neighborhood of the origin (or desirable equilibrium). This is because of the SVM model learning error. For a local asymptotical convergence, the region of attraction is a critical issue. The region of attraction may not be larger or equal to "region of operation" $\mathcal{D}$, but should be desirable large.

Problem Statement

$$\dot{x} = f(x, u), \tag{4.26}$$

where

$x \in R^n$	system states
$u \in R^m$ $(m \leq n)$	control inputs
$\dot{x}$	time derivatives of the states x
$f(\cdot) : R^{n+m} \rightarrow R^n$	unknown nonlinear function

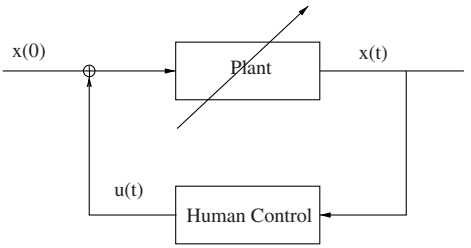


Fig. 4.6. The practical system and the human control from a dynamic system.

The control objective can be described as: find a continuous control law $u = u(x)$, such that all of the states of the above system (4.26) asymptotically tend to a bounded small neighborhood of the origin.

Assume that a practical system has been well controlled by a human expert and the all states of the system have been put into in a small neighborhood of origin. The practical system and the human control form a dynamic system shown in Figure 4.6. Usually, a continuous-time control process is approximately considered as a process of fast discrete-time sampling. This continuous-time nonlinear control system is approximately described by the difference equation of the form

$$\mathbf{x}(t+1) = f_x(\mathbf{x}(t), \mathbf{u}_h(t)) \tag{4.27}$$

where $\mathbf{x} = [x_1, x_2, ..., x_n]^T \in R^n$ is the state vector, $\mathbf{u}_h \in R^m$ is the human control vector, $f_x = [f_1, f_2, ..., f_n]^T : R^{n+m} \to R^n$ is a set of unknown nonlinear functions. We assume throughout the paper that the state vector $\mathbf{x}$ of the system can be measured.

There is a learning controller for the dynamic system (4.27), which is obtained by off-line learning from a data table produced by the human expert demonstration, shown in Figure 4.7. The difference Equation (4.28) approximately described the learning controller

$$\begin{cases} \hat{x}_1(t+1) = \hat{f}_1(\mathbf{x}(t), \mathbf{u}(t)) & = f_1(\mathbf{x}(t), \mathbf{u}(t)) + e_1(\mathbf{x}(t), \mathbf{u}(t)) \\ \hat{x}_2(t+1) = \hat{f}_2(\mathbf{x}(t), \mathbf{u}(t)) & = f_2(\mathbf{x}(t), \mathbf{u}(t)) + e_2(\mathbf{x}(t), \mathbf{u}(t)) \\ \quad\vdots & \\ \hat{x}_n(t+1) = \hat{f}_n(\mathbf{x}(t), \mathbf{u}(t)) & = f_n(\mathbf{x}(t), \mathbf{u}(t)) + e_n(\mathbf{x}(t), \mathbf{u}(t)) \\ u_1(t+1) = f_{n+1}(\mathbf{x}(t), \mathbf{u}(t)) & \\ \quad\vdots & \\ u_m(t+1) = f_{n+m}(\mathbf{x}(t), \mathbf{u}(t)), & \end{cases} \tag{4.28}$$

where $\hat{\mathbf{x}} = [\hat{x}_1, \hat{x}_2, ..., \hat{x}_n]^T \in R^n$ is the estimation for the state vector $\mathbf{x}$ and $\hat{f}_x = [\hat{f}_1, \hat{f}_2, ..., \hat{f}_n]^T : R^{n+m} \to R^n$ is an estimation for f_x. $e = [e_1, e_2, ...e_n]^T = f_x - \hat{f}_x$ is a model error. $\mathbf{u} = [u_1, u_2, ..., u_m]^T \in R^m$ is the

estimation for the human control vector $\mathbf{u}_h$ and $f_u = [f_{n+1}, f_{n+2}, ..., f_{n+m}]^T$: $R^{n+m} \rightarrow R^m$ is the estimation for next time human control.

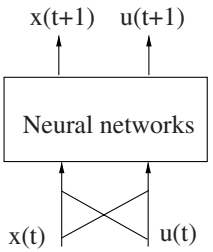


Fig. 4.7. A learning controller.

If we use the learning controller $\mathbf{u}$ to control the system, we have a new closed-form continuous-time dynamic system and it is approximately described by the difference equation of the form

$$\begin{cases} \mathbf{x}(t+1) = f_x(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{u}(t+1) = f_u(\mathbf{x}(t), \mathbf{u}(t)). \end{cases} \tag{4.29}$$

Furthermore, we let $X = [\mathbf{x}^T \ \mathbf{u}^T]^T$ and $f = [f_x^T \ f_u^T]^T$, then

$$X(t+1) = f(X(t)) \tag{4.30}$$

and let $\hat{X} = [\hat{\mathbf{x}}^T \ \mathbf{u}^T]^T$ and $\hat{f} = [\hat{f}_x^T \ f_u^T]^T$, then

$$\hat{X}(t+1) = \hat{f}(\hat{X}(t)) \tag{4.31}$$

where (4.31) is an estimation for (4.30).

The aim of this section is twofold: to formulate conditions for the system (4.30) to be stable and to determine the domain of attraction, if it is stable.

Lyapunov Theory

A very important tool in the stability analysis of discrete dynamic systems is given by Lyapunov's theory [65], [61].

Definition 1: A function $V(X)$ is said to be *positive definite* in a region $\mathcal{W}$ containing the origin if (i) $V(0) = 0$. (ii) $V[X(t)] > 0$ for all $x \in \mathcal{W}$, $X \neq 0$.

Definition 2: Let $\mathcal{W}$ be any set in R^n containing the origin and $V : R^n \rightarrow R$. We say that V is a *Lyapunov function* of Equation (4.30) on $\mathcal{W}$ if (i) V is continuous on R^n. (ii) V is positive definite with respect to the origin in $\mathcal{W}$. (iii) $\Delta V(t) \equiv V[X(t+1)] - V[X(t)] \leq 0$ along the trajectory of (4.30) for all $X \in \mathcal{W}$.

The existance of a Lyapunov function assures stability as given by the following theorem.

Theorem 1: If V is a Lyapunov function of (4.30) in some neighborhood of an equilibrium state $X = 0$, then $X = 0$ is a stable equilibrium.

If in addition $-\Delta V$ is positive definite with respect to $X = 0$, then the origin is asymptotically stable.

So far, the definition of stability and asymptotical stability are in terms of perturbations of initial conditions. If the model error e is small, one hopes that at least qualitatively, the behavior of the original system and that of the perturbed one will be similar. For the exact relation, *stable under perturbations*needs to be defined [94].

Definition 3: Let $X(X_0, t)$ denote the solution of (4.30) with the initial condition $X_0 = X(X_0, 0)$. The origin $X = 0$ is said to be *stable under perturbations* if for all $\epsilon > 0$ there exists $\delta_1(\epsilon)$ and $\delta_2(\epsilon)$ such that $||X_0|| < \delta_1$ and $||e(t, X)|| < \delta_2$ for all $k > 0$ imply $X(X_0, t) < \epsilon$ for all $t \leq 0$.

If in addition, for all ϵ there is an r and a $T(\epsilon)$ such that $||X_0|| < r$ and $||e(t, X)|| < \delta_2(\epsilon)$ for all $t > 0$ imply $||X(X_0, t)|| < \epsilon$ for all $t > T(\epsilon)$, the origin is said to be *strongly stable under perturbations*(SSUP).

Strongly stable under perturbations (SSUP) means that the equilibrium is stable, and that states started in $B_r \subset \Omega$ actually converge to the error bound at limited time. Ω is called a *domain of attraction* of the solution (while the domain of attraction refers to the largest such region, *i.e.*, to the set of all points such that trajectories initialed at these points eventually converge to the error bound.

With this in mind the following theorem [61], [94] can be stated:

Theorem 2: If f is Lipschitz continuous in a neighborhood of the equilibrium, then the system (4.30) is strongly stable under perturbations iff it is asymptotically stable.

In this paper, Support Vector Machine (SVM) will be considered as a neural network structure to learn the human expert control process. In the next section, a rough introduction to the SVM learner that we will use, will be provided.

Convergence Analysis

There are many kernels that satisfy the Mercer's condition as described in [27]. In this paper, we take a simple polynomial kernel in Equation (4.19):

$$K(X_i, X) = ((X_i \cdot X) + 1)^d, \tag{4.32}$$

where d is user defined (Taken from [108]).

After the off-line training process, we obtain the support values (α and α^*) and the corresponding support vectors. Let X_i be sample data of X. By expanding Equation (4.19) according to Equation (4.32), Let $\hat{f}(X) =$

$[\hat{f}_1, \hat{f}_2, ...\hat{f}_{m+n}]^T$ be a vector. $\hat{f}_i$ is a nonhomogeneous form of degree d in $X \in R^{n+m}$ (containing all monomials of degree $\leq d$)

$$\hat{f}_i = \sum_{0 \leq k_1+k_2+...+k_{n+m} \leq d} c_j x_1^{k_1} x_2^{k_2} ... x_n^{k_n} u_1^{k_{n+1}} ... u_m^{k_{n+m}}, \tag{4.33}$$

where $k_1, k_2, ..., k_{n+m}$ are nonnegative integers, and $c_j \in R$ are weighting coefficients. j can be $1, 2, ..., M$, where $M = \binom{n+m+d}{n+m}$. Then,

$$\hat{f}(X) = C + A'X + g(X), \tag{4.34}$$

where $C = [c_1, c_2, ...c_{m+n}]^T$ is a constant vector, $A' \in R^{(n+m)\times(m+n)}$ is a coefficient matrix for the degree 1 in X and $g(X) = [g_1(X), g_2(X), ...g_{n+m}(X)]^T$ is a vector. $g_i(X)$ is a multinomial of degree ≥ 2 of X.

Assume that we had built $m + n$ number of multiple-input-one-output SVM models. According to [108], SVM can approximate to a model in any accuracy, if the training data number is large enough, i.e. for any $\epsilon > 0$, if $\bar{n}$ is the sample data number and e is the model error, there exists a $N > 0$, such that if $\bar{n} > N$, $e < \epsilon$. The following assumptions are made for the SVM models.

Assumption 1: For system (4.30), in the region $\mathcal{D}$, the number of sample data is large enough.

Remark 3.1: Assumption 1 is usually required in control design with neural networks for function approximation [31], [53]. Then from the above analysis, we have assumption 2.

Assumption 2: For system (4.30), in the region $\mathcal{D}$, the learning precision is high.

Hence, if the model (4.31) is sufficiently accurate, according to Equation (4.34), the system (4.30) can be transformed to the Equation (4.35)

$$X(t+1) = C + A'X + g(X), \tag{4.35}$$

Since the origin is an equilibrium point of the system, we have $C = [0, 0, ...0]^T$ and then

$$X(t+1) = A'X + g(X), \tag{4.36}$$

Thus, we can use the following theorem to judge the system (4.30) is strongly stable under perturbations (SSUP) or not.

Theorem 3: For the system (4.30), with assumptions 1 and 2 being satisfied in the region $\mathcal{D}$, if $-A = (I - A')$ is a positive definite matrix and I is a $(n + m) \times (n + m)$ identical matrix, then the closed-form system (4.30) is strongly stable under perturbations (SSUP).

Proof: Let $V = X^T X$. Then, V is positive definite. By differentiating V along the trajectories of X one gets

$$\Delta V = \frac{\partial V}{\partial X} \Delta X = 2X\Delta X. \tag{4.37}$$

Substituting (4.36) and (4.37) into it, then

$$\Delta V = 2X(A' - I)X + 2Xg(X) = 2XAX + 2Xg(X).$$

If we let μ be the smallest eigenvalue of the matrix $-A$, then we have $\mu|z|^2 \leq z^T(-A)z, \quad \forall z$. The properties of $g(X)$ imply the existence of a function $\rho(X)$ such that $\lim_{X\to 0}\rho(X) = 0$ and $|g(X)| \leq \rho(X)|X|$. We can then get the estimate

$$\Delta V = 2XAX + 2Xg(X) \leq -|X|^2(\mu - 2\rho(X)).$$

The fact that ρ tends to zero as X tends to zero implies the existence of a constant $r > 0$ such that $\rho(X) < \mu/2$ whenever $|X| \leq r$. It follows that $-\Delta V$ is positive definite in $|X| < r$ and the system (4.30) is asymptotically stable. Moreover, since f is Lipschitz continuous and according to Theorem 2, we know that the system is strongly stable under perturbations (SSUP).

Remark 3.2: The technique used in the proof can in principle give an approach in estimating the domain of attraction Ω. This value of r will however often be quite conservative. The arrangement of this is a critical property to evaluate the practical value of the SVM learning controller. In the next part of this section, we will provide a better and more practical method to estimate the domain of attraction Ω.

Computation of Stability Region

The problem of determining stability regions (Subsets of domain of attraction) or robust global stability [106], of nonlinear dynamical systems is of fundamental importance in control engineering and control theory.

Let $\Delta X(t) = X(t+1) - X(t)$, we transform the system (4.36) to

$$\Delta X = AX + g(X), \tag{4.38}$$

where we assumed that A is a negative definite matrix, i.e., all eigenvalues of A have strictly negative real parts. Thus, the problem in this part is to estimate the domain of attraction of $X = 0$. The main tool in achieving this goal is the use of an appropriate Lyapunov function. In fact, there are almost an infinite kind of Lyapunov functions that can be applied for this aim. However, a large number of experiments show that the type of quadratic Lyapunov functions, which are chosen, usually can work out a desirable stability region if an SVM learning controller has fine performance in practical control experiments. In the following we will address the quadratic Lyapunov function.

$$V(X) = X^T PX, \tag{4.39}$$

where P is a positive definite symmetric $(n+m)\times(n+m)$ matrix. Assuming that the matrix

$$Q = -(A^T P + PA), \tag{4.40}$$

is positive definite, $V(X)$ is a valid Lyapunov function for the linear system. The set

$$\Omega_c = \{X | V(X) \leq c\}, \quad c > 0, \tag{4.41}$$

is contained in the unknown domain of attraction if the inequality

$$\Delta V(X) = X^T(A^T P + PA)X + 2X^T P g(X) = -X^T Q X + 2X^T P g(X) < 0, \tag{4.42}$$

is valid for all

$$X \in \Omega_c, \quad X \neq 0 \tag{4.43}$$

[66]. The problem is to maximize c, which is actually an optimization problem, such that (4.42) and (4.43) are satisfied. The corresponding set Ω_c is the largest subset of the main attraction which can be guaranteed with the chosen Lypunov function. So far, this goal can be divided into issues: to choose a suitable quadratic Lyapunov function for maximizing c and to compute c.

For the first issue, we need the following theorem [37].

Theorem 4: Let all eigenvalues of matrix A have strictly negative real parts. Then, for every symmetric positive definite matrix Q, there is a symmetric positive definite P such that

$$A^T P + PA = -Q.$$

If Q is positive definite, then so is P. The proof is in [37]. At the same time, if we choose a symmetric positive definite P, the unique Q may not be positive definite. Thus, we transfer the first issue to design a proper Q, to maximize c. Since Q is a symmetric $(n+m) \times (n+m)$ matrix, we have

$$\bar{N} = \frac{(n+m)(n+m+1)}{2} \tag{4.44}$$

independent parameters that we can design. How to choose the $\bar{N}$ parameters is open problem and there is almost no research work about it that we have found. We proposed a numerical approach to choose the $\bar{N}$ parameters to maximize c. We neglect detail of it here and go forward to the next issue problem, but have left it in the experimental part to illustrate it with an example.

The second problem can be described as if we have a Q, how to work out c. There are a number of work about the second issue problem [33], [66] and [106]. We follow the approach in [106]. If we have chosen a Q, then there is a unique symmetric and positive definite P as,

$$P = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n+m} \\ p_{1,2} & p_{2,2} & \cdots & p_{2,n+m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1,n+m} & p_{2,n+m} & \cdots & p_{n+m,n+m}. \end{pmatrix} \tag{4.45}$$

P can be efficiently decomposed by means of the Cholesky Factorization into a lower and upper triangular matrices with the following equation.

$$P = L^T L. \tag{4.46}$$

Here the matrix L is an upper triangular matrix in the form

$$L = \begin{pmatrix} l_{1,1} & l_{1,2} & l_{1,3} & \cdots & l_{1,n+m} \\ 0 & l_{2,2} & l_{2,3} & \cdots & l_{2,n+m} \\ 0 & 0 & l_{3,3} & \cdots & l_{3,n+m} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & l_{n+m,n+m}. \end{pmatrix} \tag{4.47}$$

The elements of the matrix L are calculated as follows.

$$\begin{aligned} l_{1,1} &= \sqrt{p_{1,1}}, \quad l_{1,j} = \tfrac{p_{1.,j}}{l_{1,1}}, \qquad j = 2, ..., n+m \\ l_{i,i} &= \sqrt{(p_{i,i} - \textstyle\sum_{k=1}^{i-1} l_{k,j}^2)}, \qquad j = 2, ..., n+m \\ l_{i,j} &= \tfrac{1}{l_{i,i}}(p_{i,j} - \textstyle\sum_{k=1}^{i-1} l_{k,i} l_{k,j}), \; i = 2, ..., n+m-1, \; j = i+1, ..., n+m. \end{aligned} \tag{4.48}$$

We transform the system with

$$\hat{X} = LX, \tag{4.49}$$

where $\hat{X}$ is the new state vector, into a different system, whose state space difference equations are given by the following equation

$$\Delta\hat{X} = LAL^{-1}\hat{X} + Lg(L^{-1}\hat{X}) = \hat{A}\hat{X} + \hat{g}(\hat{X}). \tag{4.50}$$

For the system, the state vector $\hat{X} = 0$ is also an equilibrium point. We can now describe the same Lyapunov function according to the new state vector $\hat{X}$,

$$V(X) = X^T PX = X^T L^T LX = \hat{X}^T\hat{X} = \hat{V}(\hat{X}). \tag{4.51}$$

According to the Lyapunov stability theory we will investigate the asymptotic stability of the system

$$\Delta\hat{X} = \hat{A}\hat{X} + \hat{g}(\hat{X}) \tag{4.52}$$

with the Lyapunov function

$$\hat{V}(\hat{X}) = 2\hat{X}^T\hat{X} = 2\hat{X}^T(\hat{A}\hat{X} + \hat{g}(\hat{X})). \tag{4.53}$$

It must be strictly negative in the domain of attraction Ω_c. We assume that the degree of $\hat{V}(\hat{X})$ is k. It is obvious that the polynomial

$$p(\hat{X}) = -\Delta\hat{V}(\hat{X}) \tag{4.54}$$

must be strictly positive in Ω_c. Furthermore, we transform the cartesian coordinates into polar coordinates with the following replacements.

$$\begin{array}{lll}
\hat{X}_1 & = r\ \cos\theta_1\ \cos\theta_2\ \cdots\ \cos\theta_{n+m-2}\ \cos\theta_{n+m-1} \\
\hat{X}_2 & = r\ \cos\theta_1\ \cos\theta_2\ \cdots\ \cos\theta_{n+m-2}\ \sin\theta_{n+m-1} \\
\hat{X}_3 & = r\ \cos\theta_1\ \cos\theta_2\ \cdots\ \cos\theta_{n+m-3}\ \sin\theta_{n+m-2} \\
\vdots & \vdots\ \vdots & \\
\hat{X}_i & = r\ \cos\theta_1\ \cos\theta_2\ \cdots\ \cos\theta_{n+m-i}\ \sin\theta_{n+m-i+1} \\
\vdots & \vdots\ \vdots & \\
\hat{X}_{n+m} & = r\ \sin\theta_1, &
\end{array} \tag{4.55}$$

where r is the radius and $\theta = [\theta_1, \theta_2, ..., \theta_{n+m-1}]^T$ are the angles. In this case

$$\hat{V}(\hat{X}) = \hat{V}(r, \theta) := \hat{V}(y) = r^2, \tag{4.56}$$

where y is the vector $[r, \theta_1, \theta_2, ..., \theta_{n+m-1}]^T$. The function $p(\hat{X})$ can be represented with the equation (4.57).

$$p(\hat{X}) = p(r, \theta) = p(y) = a_k r^k + a_{k-1} r^{k-1} + ... + a_2 r^2, \tag{4.57}$$

where $a_i,\ \ i = 2, ..., k$, is the function of the angles θ.

Next, we introduce the Theorem of Ehilich and Zeller and relative materials for working out c [106]. In the following, $J = [a, b]$ denotes a nonempty real interval with $J \subset R$. We define the set of Chebychev points at interval J for a given integer $N > 0$ by $x(N, J) := \{x_1, x_2, ..., x_N\}$, where

$$x_i := \frac{a+b}{2} + \frac{b-a}{2} \cos(\frac{(2i-1)\pi}{2N}), \quad i = 1, 2, ..., N. \tag{4.58}$$

Let p_m be the set of polynomials p in one variable with $deg\ \ p \leq m$.

Then, we extend the definitions in one variable to several variables using the following replacements. The interval J is replaced by

$$\hat{J} = [a_1, b_1] \times [a_2, b_2] \times ... \times [a_{n+m}, b_{n+m}], \tag{4.59}$$

which represents a hyperrectangle . For the degree of p with respect to the i-th variable x_i we introduce the abbreviation m_i and the set of Chebychev points in $\hat{J}$ is given by

$$x(\hat{N}, \hat{J}) := x(N_1, [a_1, b_1]) \times x(N_2, [a_2, b_2]) \times ... \times x(N_{n+m}, [a_{n+m}, b_{n+m}]), \tag{4.60}$$

where N_i is the number of Chebychev points in the interval $[a_i, b_i]$.

Using the theorem of Ehilich and Zeller, we can find out with the following inequality whether the polynomial $p(y)$ is strictly in a sphere with radius r.

$$(K+1) p_{min}^{y(\hat{N}, \hat{j})} - (K-1) p_{max}^{y(\hat{N}, \hat{j})} > 0, \tag{4.61}$$

where the angles vary in the interval $[0, 2\pi]$; the radius varies in the interval $[0, r]$;

$$p^I_{min} := min_{x\in I} p(x), \quad p^I_{max} := max_{x\in I} p(x);$$

and

$$K = \prod_{i=1}^{n+m} C(\frac{m_i}{N_i})$$

under the condition $N_i > m_i, \;\; i = 1, 2, ..., n+m$ and $C(q) = [\cos(\frac{q}{2}\pi)]^{-1}$ for $0 < q < 1$. If the inequality (4.61) is valid, the following inequality are also valid.

$$(K+1)p(y[i]) - (K-1)p(y[j]) > 0, \quad i, j = 1, 2, ..., \hat{N} \tag{4.62}$$

with

$$p^{y(\hat{N},\hat{j})}_{min} \le p(y[i]) \le p^{y(\hat{N})}_{max}, \quad i = 1, 2, ..., j, ..., \hat{N} \tag{4.63}$$

where $y[i], y[j] \in y(\hat{N}, \hat{j})$ are two Chebychev points. For $\hat{N}$ Chebychev points we have $\hat{N}^2$ inequalities of type (4.62) which are equivalent to (4.61). (4.62) provide us the sufficient conditions for the strict positive of polynomial $p(y)$.

Moreover,

$$p(r, \theta[i]) > 0, \quad i = 1, 2, ..., \hat{N} \tag{4.64}$$

give us the necessary conditions for the strict positive of polynomial $p(y)$.

If the inequalities (4.62) are numerically solved, an inner approximation r^*_{in} to the maximum radius r^* is determined. The solution of inequalities (4.64) give us an outer approximation r^*_{out} to r^*. In this case

$$r^*_{in} \le r^* \le r^*_{out} \tag{4.65}$$

is valid. The maximum level of the surface c^* is equal to $(r^*)^2$ and gives the set Ω_{c^*}. c^* lies

$$c^*_{in} \le c^* \le c^*_{out} \tag{4.66}$$

and Ω_{c^*} is the largest subset of the domain of attraction with a given Lyapunov function. Now, we will provide the experimental work as an example to illustrate the algorithm.

4.2.4 Experiments

Experimental System – Gyrover

The single-wheel gyroscopically-stabilized robot, Gyrover, takes advantage of the dynamic stability of a single wheel. Figure 4.8 shows a photograph of the third Gyrover prototype.

Gyrover is a sharp-edged wheel with an actuation mechanism fitted inside the rim. The actuation mechanism consists of three separate actuators: (1) a spin motor, which spins a suspended flywheel at a high rate and imparts dynamic stability to the robot; (2) a tilt motor, which steers the Gyrover;

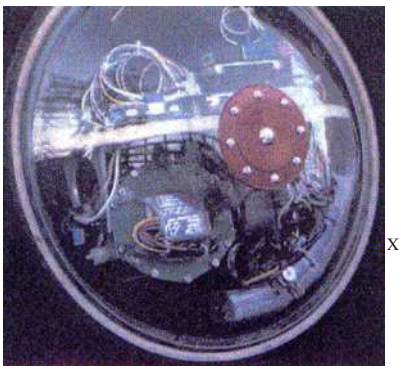


Fig. 4.8. Gyrover: A single-wheel robot.

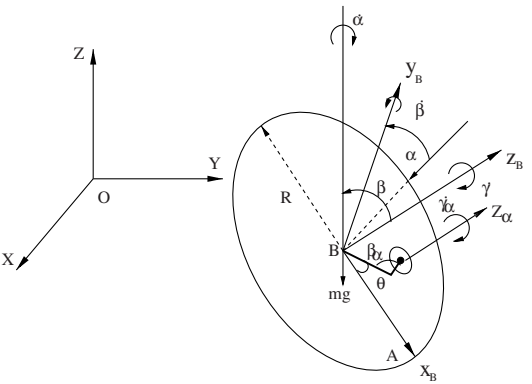


Fig. 4.9. Definition of the Gyrover's system parameters.

and (3) a drive motor, which causes forward and/or backward acceleration by driving the single wheel directly.

The Gyrover is a single-wheel mobile robot that is dynamically stabilizable but statically unstable. As a mobile robot, it has inherent nonholonomic constraints. First-order nonholonomic constraints include constraints at joint velocities and Cartesian space velocities. Because no actuator can be used directly for stabilization in the lateral direction, it is an underactuated nonlinear system. This gives rise to a second-order nonholonomic constraint as a consequence of dynamic constraints introduced by accelerative forces on passive joints.

To represent the dynamics of the Gyrover, we need to define the coordinate frames: three for position (X, Y, Z), and three for the single-wheel orientation (α, β, γ). The Euler angles (α, β, γ) represent the precession, lean and rolling angles of the wheel respectively. (β_a, γ_a) represent the lean and rolling angles of the flywheel respectively. They are illustrated in Figure 4.9.

Task and Experimental Description

The aim of this experiment is twofold: to compare the ability of an SVM with several general ANNs in learning human control skills and to train an SVM learning controller, test it with Gyrover to illustrate the applications of the previous asymptotically stability analysis and verify it.

The control problem consists of controlling Gyrover in keeping it balanced, i.e., not falling down on the ground. We will build up an SVM learning controller based on learning imparted from expert human demonstrations.

There are mainly two major control inputs: U_0 controlling the rolling speed of the single wheel $\dot{\gamma}$, and U_1 controlling the angular position of the flywheel

β_a. For the manual-model (i.e., controlled by a human), U_0 and U_1 are input by joysticks, and in the auto-model they are derived from the software controller. During all experiments, we only used of U_1. β and β_a among the state variables are used during the training process. In order to construct a controller for balance, the trained model is adjusted in calibration with the above state variables, and its output U_1.

A human expert controlled Gyrover to keep it balanced and produced around 2,400 training samples. Table 4.2 displays some raw sensor data from the human expert control process.

Table 4.2. Sample human control data.

Input		Output
β	β_a	U_1
5.5034	2.4023	179.0000
5.7185	2.3657	176.0000
5.6012	2.1313	170.0000
5.1271	2.1460	170.0000
5.9433	1.0425	143.0000

After calibrating the data to the original points, we put the sample data into two groups as training data table and testing data table to train an SVM learning controller. Vapnik's Polynomial kernel of order 2 in Equation (4.32) is chosen and the input vector consists of current state variables (β_a, β,). The output consists of control input U_1. Three-input-one-output SVM models for each of the three variables were trained with the three current values as inputs. For $\hat{\beta}_a(t+1)$, $\hat{\beta}(t+1)$, and $u(t+1)$ we have 977, 989 and 905 support vectors, respectively with corresponding α_i and α_i^*. By expanding the three SVM models according to Equation (4.32) we have the following Equation

$$X(t+1) = A'X + g(X), \tag{4.67}$$

where $X(t+1) = [\hat{\beta}_a(t+1)\ \hat{\beta}(t+1)\ u(t+1)]^T = [x_1, x_2, x_3]^t$,

$$A' = \begin{bmatrix} 0.8818 & 0.0074 & -0.2339 \\ -0.1808 & 0.8615 & -0.2389 \\ 0.0154 & -0.0007 & 1.0167 \end{bmatrix} \tag{4.68}$$

and

$$g(X) = \begin{bmatrix} 0.0004x_1^2 - 0.0013x_1x_2 + 0.0028x_1x_3 - 0.0017x_2x_3 - 0.0006x_2^2 + 0.0027x_3^2 \\ 0.0004x_1^2 + 0.0002x_1x_2 + 0.0034x_1x_3 + 0.0017x_2x_3 + 0.0003x_2^2 + 0.0049x_3^2 \\ 0.0002x_1^2 + 0.0002x_1x_2 - 0.0002x_1x_3 - 0.0001x_2x_3 - 0.0001x_2^2 - 0.0007x_3^2 \end{bmatrix} \tag{4.69}$$

Then, we have $A = A' - I$, which is a negative definite matrix. Thus, the system is strongly stable under perturbations (SSUP).

Next, we need to estimate the domain of attraction Ω. If we use $V = X^T X$ and the Lyapunov function, i.e., Let P be an identical matrix, by application of the algorithm in inequalities (4.62) and (4.64), we have $r_i^* n = 1.991$ and $r_o^* ut = 2.005$, then $r^* \approx 2$. It is too small for a lean angle β in an interval of $[-2, 2]$ degrees.

Then, since $m + n = 2 + 1 = 3$, from Equation (4.44), for Q, we have 6 independent parameters that we can adjust. By some interval

$$q_i = [a_i, b_i], \quad i = 1, 2, ..., 5 \tag{4.70}$$

we find out a suitable Q

$$Q = \begin{bmatrix} 0.1 & 0 & 0.1877 \\ 0 & 0.4 & 0.0329 \\ 0.1877 & 0.0329 & 1 \end{bmatrix}. \tag{4.71}$$

From Equation (4.40) we have

$$P = \begin{bmatrix} 2.67141 & -0.955202 & 6.05634 \\ 6.05634 & 1.3957 & -0.955202 \\ -0.542122 & -0.542122 & 47.1831 \end{bmatrix}. \tag{4.72}$$

If we use $V = X^T PX$ and the Lyapunov function, by application of the algorithm in inequalities (4.62) and (4.64), we have $r_i^* n = 21.9$ and $r_o^* ut = 22.0$, then $r^* \approx 22$. The lean angle β is in a region of $[-22, 22]$ degrees, which is a suitable range that the lean angle can reach in a balanced experiment.

Experimental Results

By using the SVM learning controller to execute the balance experiment, the experiment is successful. But by using a Backpropagation neural network and the radial basis function(RBF) neural network to train the learning controller with the same data table, the result failure. Figure 4.10 shows the tilt angle β_a, lean angle β of SVM learning control, and Figure 4.11 shows the comparison of the same Human control and SVM learner.

4.3 Learning Control with Limited Training Data

Researchers have become increasingly interested in using artificial neural networks (ANN) for learning control [4], [36], [37], [73] and [119]. A learning controller, here, consists of a neural network model that has learned one or more sets of system state and control input data from human expert demonstrations and then, in the control process, uses the neural network model to produce system control inputs according to the current system states. Learning control can be considered as a process of building a mapping between system states and control inputs, i.e., a function regression problem.

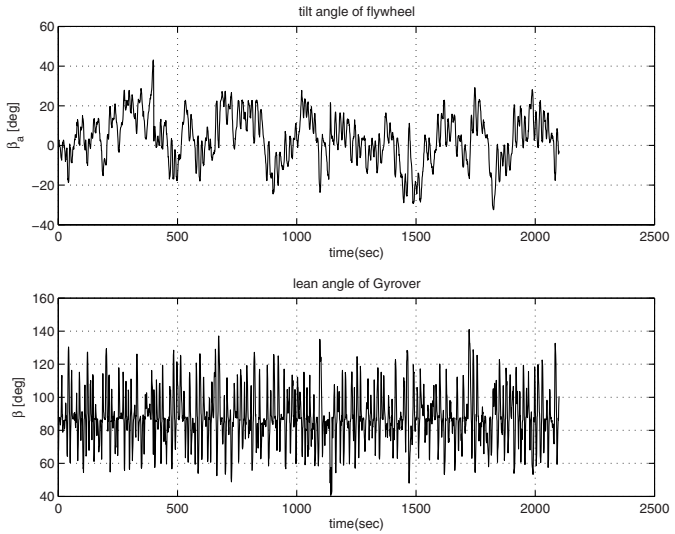


Fig. 4.10. The tilt angle β_a Lean angle β of SVM learning control.

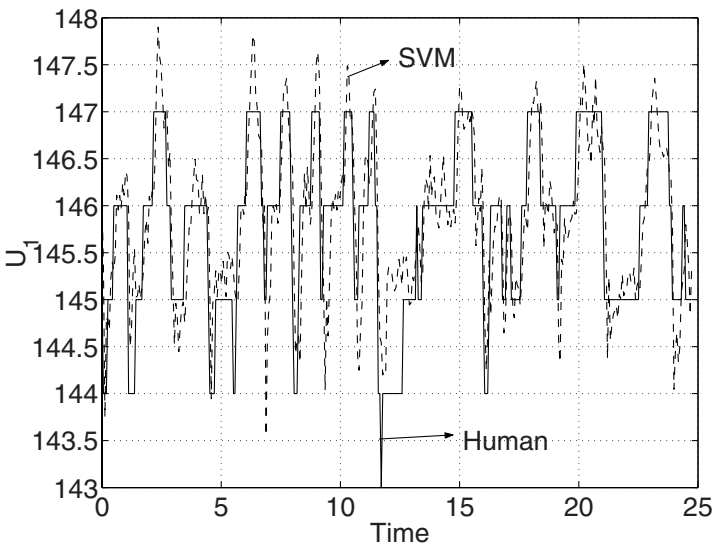


Fig. 4.11. U_1 comparison of the same Human control and SVM learner.

It is well known that a finite number of training samples causes practical difficulties and constraints in realizing good learning [83]. Raudys and Jain [84] point out that ANNs will also encounter similar difficulties and constraints when the ratio of training sample size to input dimensionality is small. In

general, for applications with a large number of features and complex learning rules, the training sample size must be quite large. A large test sample set is particularly essential to accurately evaluate a learner with a very low error rate. In order to understand this important effect, consider the following very simple technique (not recommended in practice) for modelling non-linear mapping from a set of input variables $\mathbf{x}$ to an output variable y on the basis of a set of training data.

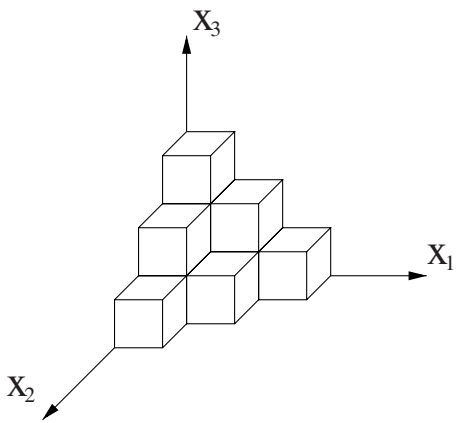


Fig. 4.12. Curse of dimensionality.

We begin by dividing each of the input variables into a number of intervals, so that the value of a variable can be specified approximately by laying in which number of boxes or cells as indicated in Figure 4.12. Each of the training examples corresponds to a point in one of the cells, and carries an associated value of the output variable y. If we are given a new point in the input space, we can determine a corresponding value for y by finding which cell the point falls in, and then returning the average value of y for all of the training points which lie in that cell. By increasing the number of divisions along each axis we could increase the precision with which the input variables can be specified. There is, however, a major problem. If each input variable is divided into M divisions, then the total number of cells is M^d and grows *exponentially* with the dimensionality of the input space. Since each cell must contain at least one data point, this implies that the quantity of training data needed to specify the mapping also grows exponentially. This phenomenon has been termed the *curse of dimensionality* [11]. If we are forced to work with a limited quantity of data, then increasing the dimensionality of the space rapidly leads to the point where the data is very sparse, in which case it provides a very poor representation of the mapping.

A large number of training samples can be very expensive and time consuming to acquire, especially, during an on-line learning control process. For

instance, we can only collect very limited training samples from a cycle of human demonstration. It is well known that when the ratio of the number of training samples to the VC (*Vapnik-Chervonenkis*) dimension of the function is small, the estimates of the regression function are not accurate and, therefore, the learning control results may not be satisfactory. Meanwhile, the real-time sensor data always have random noise. This has a bad effect on the learning control performance, as well. Thus, we need large sets of data to overcome these problems. Moreover, sometimes we need to include some history information of systems states and/or control inputs into the ANN inputs to build a more stable model. These will cause the increasing of the dimension or features of the neural network and increase the requirement for more training data.

In this work, our main aim is to produce more new training samples (called unlabelled sample, here) without increasing costs and to enforce the learning effect, so as to improve learning control.

The main problem in statistical pattern recognition is to design a classifier. A considerable amount of effort has been devoted to designing a classifier in small training sample size situations [40], [41] and [83]. Many methods and theoretical analysis have focused on nearest neighbor re-sampling or bootstrap re-sampling. However, the major problem in learning control is function approximation. There is limited research exploring function regression under conditions of sparse data. Janet and Alice's work in [18] examined three re-sampling methods (cross validation, jackknife and bootstrap) for function estimation.

In this chapter, we use the local polynomial fitting approach to individually rebuild the time-variant functions of system states. Then, through interpolation in a smaller sampling time interval, we can rebuild any number of new samples (or unlabelled samples).

4.3.1 Effect of Small Training Sample Size

Our learning human control problem might be thought of as building a map between the system states X and the control inputs Y, approximately. Both $X = (x^1, x^2, ...x^m)$ and Y are continuous time-various vectors, where x_i is one of the system states and they are true values, but not random variables. In fact, X may consist of a number of variables in previous and current system states and previous control inputs. Y are current control inputs. Furthermore, without the loss of generality, we restrict Y to be a scalar for the purposes of simplifying the discussion.

$$Y = \hat{F}(X) + Err, \tag{4.73}$$

where $\hat{F}$ is the estimation for true relation function F and $Err \in R$ is the total error of the learning, which need to be reduced to lower than some desirable value.

In this problem, we mainly face three types of error. The first type of error is measurement error, which is derived from the sensor reading process. We call it observation error. It may exist in both X and Y. The second type of error occurs due to the difference between the true mapping function and some fixed neural network structure. We call it structure error. As proven by Cybenko [28] in 1989, most functions (including any continuous function with bounded support) can be approximated by functions of a one-hidden-layer BP neural network. Thus, we omit this type of error in our following discussion and assume that we already have a good enough structure that allows us to approximate any function, if we choose suitable parameters for that structure, i.e. there exists θ^*, such that

$$Y = F(X) = f(X, \theta^*), \tag{4.74}$$

where $\theta^* = (\theta_1^*, \theta_2^*, ..., \theta_h^*)^T$, h is the number of the structure parameters. The last type of error is the parameter error and it is the result of the estimating of the best suitable parameters. The three types of error are closely related to the estimation error. To compensate for the observation error, more training data lead to better results. At the same time, the parameter error also requires more training data to overcome the *overfitting* problem, which will be discussed in more detail in next section. Thus, the estimation error can be reduced, but not removed absolutely, because it is impossible to collect infinite training samples.

Next, we will address our problem by considering these errors, individually. Let Y^o be the observations for true Y. Let

$$Y^o = Y + \epsilon,$$

where ϵ has the Gaussian distribution with $E(\epsilon) = 0$ and variance $\sigma^2 > 0$. If X^o be the observation of X,

$$X^o = X + \varepsilon,$$

where $\varepsilon \in R^m$. Here, we assume each system state x_i has the similar observation error as Y. The distribution of ε has the form $N(0, \sigma^2 \mathbf{I}_m)$. Here, we have a data table, which includes the observation for both inputs and outputs to train a neural network as a learning controller. The data table comes from discrete time sampling from human expert control demonstrations. In fact, from the data table, we have Y^o and X^o, but not Y and X, thus we modify Equation (4.74) to

$$Y^o = f(X, \theta^*) + \epsilon. \tag{4.75}$$

Usually, observation errors are much smaller then their true value. A Taylor expansion to the first order can be used to approximate $f(X^o, \theta)$ in terms of $f(X, \theta)$

$$f(X^o, \theta) \approx f(X, \theta) + (L_X f)^T \cdot \varepsilon, \tag{4.76}$$

where $L_X f = (\frac{\partial f(X,\theta)}{\partial x^1}, \frac{\partial f(X,\theta)}{\partial x^2}, ..., \frac{\partial f(X,\theta)}{\partial x^m})^T$.

We focus our attention on the estimation of regression function f, here, for θ^*. We show that for a fairly broad class of sampling scheme t_i with a fixed and identical sampling rate $\Delta > 0$. Assume that $(\hat{f})$ is an estimation of Y from a random sample T of size n. The system is represented by Equation (4.75). The least-squares estimate of θ^* is $\hat{\theta}$, which is obtained by minimizing the error function given in (4.77) (for neural networks, the error exappropriation algorithm is a common method for minimizing the error function). The predicted output from the model is $\hat{Y}^o$, as shown in (4.78)

$$S(\theta) = \sum_{i=1}^{n} [Y_i^o - f(X_i^o, \theta)]^2 \tag{4.77}$$

$$\begin{aligned} \hat{Y}_i^o &= f(X_i^o, \hat{\theta}) \\ &= f(X_i, \hat{\theta}) + (L_X f)^T \cdot \varepsilon. \end{aligned} \tag{4.78}$$

If the model provides an accurate prediction of the actual system behavior, then $\hat{\theta}$ is close to the true value of the set of parameters θ^* and a Taylor expansion to the first-order can be used to approximate $f(X_i, \hat{\theta})$ in terms of $f(X_i, \theta^*)$

$$f(X_i, \hat{\theta}) \approx f(X_i, \theta^*) + (L_\theta f)^T \cdot (\hat{\theta} - \theta^*),$$

where $L_\theta f = (\frac{\partial f(X,\theta^*)}{\partial \theta_1^*}, \frac{\partial f(X,\theta^*)}{\partial \theta_2^*}, ..., \frac{\partial f(X,\theta^*)}{\partial \theta_h^*})^T$. Then, Equation (4.78) turns to be

$$\hat{Y}_i^o \approx f(X_i, \theta^*) + (L_\theta f)^T \cdot (\hat{\theta} - \theta^*) + (L_X f)^T \cdot \varepsilon. \tag{4.79}$$

The subscript value of o is given to denote the set of observation points other than that used for the least-squares estimation of θ^*. By ignoring the second-order small quantity and using Equations (4.75) and (4.79), Equation (4.80) gives the difference between the new observation Y_0^o and the predicted value $\hat{Y}_0^o$, and Equation (4.81) gives the expected value of the difference.

$$Y_0^o - \hat{Y}_0^o \approx \epsilon_0 - (L_\theta f_0)^T (\hat{\theta} - \theta^*) - (L_X f_0)^T \cdot \varepsilon_0. \tag{4.80}$$

$$E[Y_0^o - \hat{Y}_0^o] \approx E[\epsilon_0] - (L_\theta f_0)^T E[(\hat{\theta} - \theta^*)] - (L_X f_0)^T E[\varepsilon_0] \approx 0. \tag{4.81}$$

Because of the statistical independence among ϵ_0, $\hat{\theta}$ and ε_0, the variance can be expressed as

$$var[Y_0^o - \hat{Y}_0^o] \approx var[\epsilon_0] + var[(L_\theta f_0)^T (\hat{\theta} - \theta^*)] + var[(L_X f_0)^T \varepsilon_0]. \tag{4.82}$$

The distribution of $(\hat{\theta} - \theta^*)$ can be approximated as having the distribution $N(0, \sigma_\theta^2 [\mathbf{F}_\cdot(\hat{\theta})^T \mathbf{F}_\cdot(\hat{\theta})]^{-1})$ in [24]. The Jacobian matrix $\mathbf{F}_\cdot(\hat{\theta})$ has the form shown in Equation (4.83), where the single period is placed to accord with the notations used in [93] which denotes that the matrix has first-order differential terms

$$\mathbf{F}_{\cdot}(\theta^*) = \begin{bmatrix} (\frac{\partial f(X_1,\theta^*)}{\partial \theta_1^*}) & (\frac{\partial f(X_1,\theta^*)}{\partial \theta_2^*}) & \cdots & (\frac{\partial f(X_1,\theta^*)}{\partial \theta_h^*}) \\ (\frac{\partial f(X_2,\theta^*)}{\partial \theta_1^*}) & \cdots & \cdots & (\frac{\partial f(X_2,\theta^*)}{\partial \theta_h^*}) \\ \vdots & \vdots & \vdots & \vdots \\ (\frac{\partial f(X_n,\theta^*)}{\partial \theta_1^*}) & (\frac{\partial f(X_n,\theta^*)}{\partial \theta_2^*}) & \cdots & (\frac{\partial f(X_n,\theta^*)}{\partial \theta_h^*}) \end{bmatrix} \tag{4.83}$$

$$var[Y_0^o - \hat{Y}_0^o] \approx \sigma^2 + \sigma_\theta^2 (L_\theta f_0)^T (\mathbf{F}_{\cdot}^T \mathbf{F}_{\cdot})^{-1} (L_\theta f_0) + \sigma^2 (L_X f_0)^T (L_X f_0)]. \tag{4.84}$$

The matrix $\mathbf{F}$ has the dimensions n by h, where n is the number of samples and h is the number of parameters θ, which composes $\hat{\theta}$. To find the effects of more sampling data on $var[Y_0^o - \hat{Y}_0^o]$, we need the following useful result.

Lemma 1 Let A and D be nonsingular squared matrices of orders k. Then, provided that the inverses exist,

$$(A + D)^{-1} = A^{-1} - A^{-1}(D^{-1} + A^{-1})^{-1} A^{-1}. \tag{4.85}$$

Proof: A verification is possible by showing that the product of $(A+D)^{-1}$ and the right-hand side of Equation (4.85) is the identity matrix.

If we have more training samples $\acute{n} = n + n_1$, let

$$\acute{\mathbf{F}}_{\cdot}(\theta^*) = \begin{bmatrix} (\frac{\partial f(X_1,\theta^*)}{\partial \theta_1^*}) & (\frac{\partial f(X_1,\theta^*)}{\partial \theta_2^*}) & \cdots & (\frac{\partial f(X_1,\theta^*)}{\partial \theta_h^*}) \\ (\frac{\partial f(X_2,\theta^*)}{\partial \theta_1^*}) & \cdots & \cdots & (\frac{\partial f(X_2,\theta^*)}{\partial \theta_h^*}) \\ \vdots & \vdots & \vdots & \vdots \\ (\frac{\partial f(X_{\acute{n}},\theta^*)}{\partial \theta_1^*}) & (\frac{\partial f(X_{\acute{n}},\theta^*)}{\partial \theta_2^*}) & \cdots & (\frac{\partial f(X_{\acute{n}},\theta^*)}{\partial \theta_h^*}). \end{bmatrix} \tag{4.86}$$

The matrix $\acute{\mathbf{F}}$ has the dimensions $\acute{n}$ by h. Then,

$$\acute{\mathbf{F}}^T \acute{\mathbf{F}} = \mathbf{F}^T \mathbf{F} + \mathbf{F}_1^T \mathbf{F}_1, \tag{4.87}$$

where $\mathbf{F}_1$ is similarly defined as in Equation (4.83), but n is substituted with n_1. Let X be a $h \times 1$ vector. Since $\mathbf{F}^T\mathbf{F}$ and $\mathbf{F}_1^T\mathbf{F}_1$ are positive definite matrices, substituting $A = \mathbf{F}^T\mathbf{F}$, and $D = \mathbf{F}_1^T\mathbf{F}_1$, in Equation (4.85), we have A^{-1} and $A^{-1}(D^{-1} + A^{-1})^{-1}A^{-1}$ are positive definite, provided that they exist. From **Lemma 1**,

$$X^T (A + D)^{-1} X \leq X^T A^{-1} X. \tag{4.88}$$

Then, from Equation (4.87)

$$X^T (\acute{\mathbf{F}}^T \acute{\mathbf{F}})^{-1} X \leq X^T (\mathbf{F}^T \mathbf{F})^{-1} X. \tag{4.89}$$

Hence, when h is fixed, the larger is n, the smaller is $var[Y_0^o - \hat{Y}_0^o]$. However, for our problem, that n is not large enough, thus, the major errors come from the second term of Equation (4.84). In this chapter, we propose to use unlabelled training samples to reduce the variance of Y, i.e., to decrease the learning error. The methods will be addressed in following sections in greater detail. If we define the data in the data table (which has been collected from

human demonstrations) as *labelled* data, we then call the new data *unlabelled* data, which are produced by interpolation from the labelled data. Let $(\acute{X}, \acute{Y})$ be one unlabelled data point.

If we assumed that the interpolation process is an unbiased estimation, then let

$$\acute{X} = X + \acute{\varepsilon}, \quad \acute{Y} = Y + \acute{\epsilon}, \tag{4.90}$$

where the distributions of $\acute{\varepsilon}$ and $\acute{\epsilon}$ have the form $N(0, \acute{\sigma}^2 \mathbf{I}_m)$ and $N(0, \acute{\sigma}^2)$. Since the unlabelled data are produced from original observation data (X^o, Y^o), they will have larger errors than the original data, i.e., $\acute{\sigma}^2 > \sigma^2$. Both the sensor reading error and the independent interpolation error are correlative with each other. Assume that totally $\acute{n}(> n)$ unlabelled data points are produced. If the unlabelled data are used to train the neural network, we need to update Equations (4.75) - (4.84) with Equation (4.90). Finally, we have

$$var[Y_0^o - \hat{Y}_0^o] \approx \acute{\sigma}^2 + \sigma_\theta^2 (L_\theta f_0)^T (\acute{\mathbf{F}}_\cdot^T \acute{\mathbf{F}}_\cdot)^{-1} (L_\theta f_0) + \acute{\sigma}^2 (L_X f_0)^T (L_X f_0)], \tag{4.91}$$

where $\acute{\mathbf{F}}_\cdot$ is defined in Equation (4.86) and $\acute{n}$ is the number of unlabelled samples used to obtain θ. Since $\acute{n} > n$, we reduce the second term of $var[Y_0^o - \hat{Y}_0^o]$. However, the first term and third term are not relative to the sample number n, but the accuracy of the training sample. In fact, by using the unlabelled training data we increased them. The final result of $var[Y_0^o - \hat{Y}_0^o]$ is dependent on the competition of the three terms. In most cases, especially, when the original training data is very sparse, the second term in Equation (4.91) dominates the learning error. Thus, by using unlabelled data, we can reduce the learning error. However, when the number of unlabelled data continue to increase, the effect of the other two terms will be larger than the second term, finally. Moreover, the interpolation error may also be larger, i.e., $\acute{\sigma}$ will increase. This is because of the *overfitting* phenomenon. Thus, using too much unlabelled data will not necessarily lead to better learning results. Later simulations will also demonstrate this point.

The Overfitting Phenomenon

Here, we choose neural networks to realize the function regression. Many of the important issues concerning the application of neural networks can be introduced in the simpler context of polynomial curve fitting. The problem is to fit a polynomial to a set of N data points by a technique of minimizing error function. Consider the Mth-order polynomial given by

$$f(x) = w_0 + w_1 x + ... + w_M x^M = \sum_{j=0}^{M} w_j x^j. \tag{4.92}$$

This can be regarded as a non-linear mapping which takes x as input and produces $f(x)$ as output. The precise form of the function $f(x)$ is determined by

the values of the parameters $w_0, ..., w_M$, which are analogous to the weights in a neural network. It is convenient to denote the set of parameters $(w_0, ..., w_M)$ by the vector W. The polynomial can then be written as a functional mapping in the form $f = f(x, W)$.

We shall label the data with the index $i = 1, ..., n$, so that each data point consists of a value of x, denoted by x_i, and a corresponding desired value for the output f, which we shall denote by y_i.

Arti Scholkopf [15] pointed out that the *actual risk* $R(W)$ of the learning machine is expressed as:

$$R(W) = \int \frac{1}{2} \| f(\mathrm{x}, W) - y \| dP(\mathrm{x}, y). \tag{4.93}$$

The problem is that $R(W)$ is unknown, since $P(\mathrm{x}, y)$ is unknown.

The straightforward approach to minimize the *empirical risk*,

$$R_{emp}(W) = \frac{1}{n} \sum_{i=1}^{l} \frac{1}{2} \| f(\mathrm{x}, W)_i - y_i \|,$$

turns out not to guarantee a small actual risk $R(w)$, if the number n of training examples is limited. In other words: a small error on the training set does not necessarily imply a high *generalization* ability (i.e., small error on an independent test set). This phenomenon is often referred to as *overfitting*. For the learning problem, the *Structure Risk Minimization* (SRM) principle is based on the fact that for any $W \in \Lambda$ and $n > h$, with a probability of at least 1-η, the bound

$$R(W) \leq R_{emp}(W) + \Phi(\frac{h}{n}, \frac{log(\eta)}{n}) \tag{4.94}$$

holds, where the *confidence term* Φ is defined as

$$\Phi(\frac{h}{n}, \frac{log(\eta)}{n}) = \sqrt{\frac{h(log\frac{2n}{h} + 1) - log(\eta/4)}{n}}.$$

The parameter h is called the VC (*Vapnik-Chervonenkis*) dimension of a set of functions, which describes the capacity of a set of functions. Usually, to decrease the $R_{emp}(w)$ to some bound, most ANNs with complex mathematical structures have a very high value of h. It is noted that when n/h is small (for example less than 20, the training sample is small in size), Φ has a large value. When this occurs, performance poorly represents $R(W)$ with $R_{emp}(W)$. As a result, according to the SRM principle, a large training sample size is required to acquire a satisfactory learning machine.

We can illustrate this by a one-dimension polynomial curve fitting problem. Assume we generate training data from the function

$$h(x) = 0.5 + 0.4 \sin(2\pi x) + \epsilon, \tag{4.95}$$

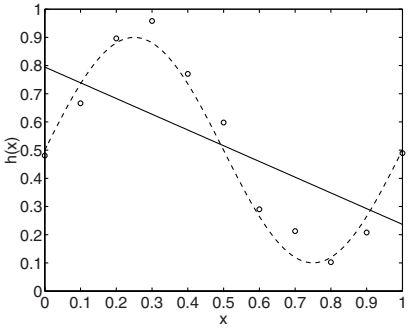


Fig. 4.13. Linear regression M=1.

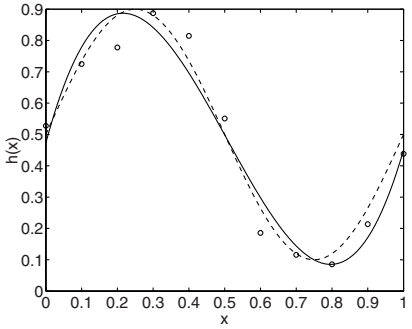


Fig. 4.14. Polynomial degree M=3.

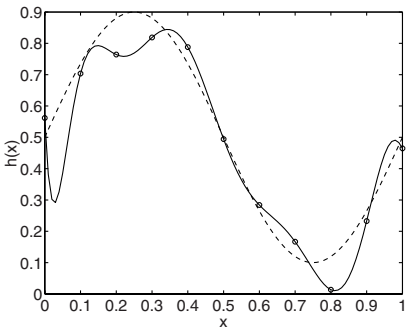


Fig. 4.15. Polynomial degree M=10.

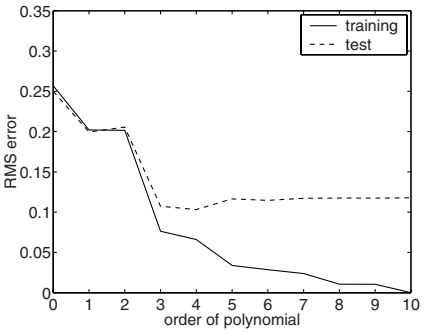


Fig. 4.16. The RMS error for both training and test sets.

by sampling the function $h(x)$ at an equal interval of x and then adding random noise with Gaussian distribution having a standard deviation $\sigma = 0.05$. We will therefore be interested in seeing how close the polynomial $f(x, W)$ is to the function $h(x)$. Figure 4.13 shows the 11 points from the training set, as well as the function $h(x)$, together with the result of fitting a linear polynomial, given by (4.92) with $M = 1$. As can be seen, this polynomial gives a poor representation of $h(x)$. We can obtain a better fit by increasing the order of the polynomial, since this increases the number of the free parameter number in the function, which gives it greater flexibility. Figure 4.14 shows the result of fitting a cubic polynomial ($M = 3$), which gives a much better approximation to $h(x)$. If, however, we increase the order of the polynomial too far, then the proximation to the underlying function actually gets worse. Figure 4.15 shows the result of fitting a 10th-order polynomial ($M = 10$). This is now able to achieve a perfect fit to the training data. However, the polynomial has fitted the data by developing some dramatic oscillations. Such functions are said to be *over-fitting* to the data. As a consequence, this function gives a poor representation of $h(x)$. Figure 4.16 shows a plot of R_{emp} for both the training data set and the test data set, as a function of order

M of the polynomial. We see that the training set error decreases steadily as the order of the polynomial increases. The test set error, however, reaches a minimum at $M = 3$, and thereafter increases as the order of the polynomial is increased.

4.3.2 Resampling Approach

Our purpose is to improve the function estimation performance of ANN learning controllers by using the polynomial fitting approach interpolation samples. Let

$$x_i^T = \{x_i(t_1), x_i(t_2), ..., x_i(t_n)\}$$

be an original training sample set of one system state, by the same sampling rate $\Delta > 0$ and $T_j = (X_j, Y_j)$, where $X_j = \{x_1(t_j), x_2(t_j), ..., x_m(t_j)\}$ is an original training sample point. An unlabelled sample set $\bar{T} = \{\bar{T}_1, \bar{T}_2, ...\bar{T}_N\}$ which has the size of $\acute{n}$ can be generated by the following.

Step 1 Using the original training sample set x_1^T to produce the segment of local polynomial estimation $\hat{x}_1(t)$ of $x_1(t)$ with respect to time $t \in (t_1, t_n)$. Let

$$x_1(t) = \hat{x}_1(t) + \acute{\epsilon},$$

where $\hat{x}_1, x_1, t \in R$

Step 2 Repeat step 1 to produce the local polynomial estimation $\hat{x}_i(t)$ $(i = 2, 3, ..., m)$ and $\hat{Y}(t)$ of $x_i(t)$ $(i = 2, 3, ..., m)$ and $Y(t)$ with respect to time $t \in (t_1, t_n)$.

Step 3 Divide the sampling rate $\Delta > 0$ by $k = \acute{n}/n + 1$ to produce new sampling time interval $\Delta_k = \Delta/k$. We produce new unlabelled samples $\acute{T}$ by a combination of interpolating the polynomials $\hat{x}_i(t)$ $(i = 1, 2, ..., m)$ and $\hat{Y}(t)$ in the new sampling rate.

Figure 4.17 shows an example of one unlabelled training sample generation process. In our method, unlabelled training samples can be generated in any number.

Barron [8] has studied the way in which the residual sum-of-squares error decreases as the number of parameters in a model is increased. For neural networks he showed that this error falls as $O(1/M')$, where M' is the number of hidden nodes in a one hidden layer network. By contrast, the error only decreases as $O(1/M^{2/d})$, where d is the dimensionality of the input space for polynomials, or indeed any other series expansion in which it is the coefficients of linear combinations of fixed functions that are adapted. However, from the former analysis, the number of hidden nodes M' chosen cannot be arbitrary large. For a practitioner, his theory provides the guidance to choose the number of variables d, the number of network nodes M', and the sample size n, such that both $1/M'$ and $(M'd/n)\log n$ are small. In particular, with $M' \sim (n/(d\log n))^{1/2}$, the bound on the mean squared error is a constant multiple of $(d\log n/n)^{1/2}$.

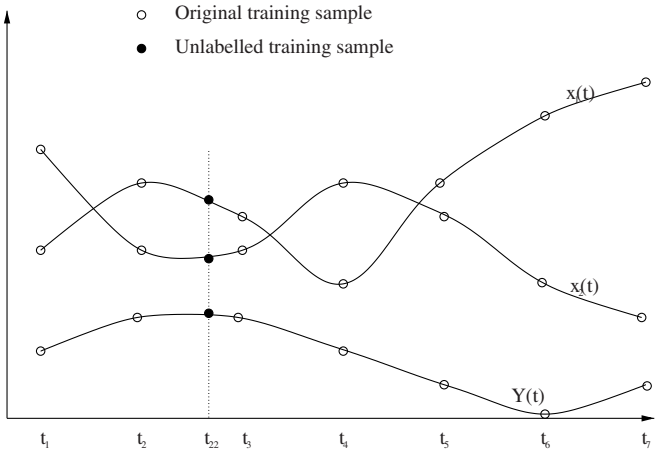


Fig. 4.17. Examples of the unlabelled sample generation, when $k = 3$.

Unlabelled training sample data can be produced in any large number by interpolation. Provided this condition, if we choose some suitable and large number of nodes (M'), we can obtain a very high learning precision. At this time, the learning error is mainly derived from the polynomial curve fitting, instead of neural network learning. However, we have improved the learning by these original limited training samples, for the learning problem has turned the inputs from $x \in R^d$ to $t \in R^1$. By using this approach, we have turned a multivariate function estimation to a univariate function estimation (i.e. one-by-one mapping).

From Equation (4.91), the estimation accuracy of this interpolation process is important, for it will affect the first and third terms in Equation (4.91) and finally, the learning controller. In the next section, the local polynomial fitting algorithm for one dimensional polynomial curve fitting is introduced as a further improvement.

4.3.3 Local Polynomial Fitting (LPF)

Advantages of Local Polynomial Fitting

Compared with the traditional method of polynomial curve fitting, there are three main advantages in using the LPF method.

First, polynomial curve fitting is a parametric regression method, based on an important assumption that the estimation model is correct. Otherwise, a large bias will inhibit precision. LPF as a nonparametric regression approach removes this assumption. Thus, the approach can be feasible for any model in unknown form and still retain a uniform convergence.

Second, LPF is a well-defined approach for function estimation, which is based on rigorous theoretical research. To obtain an ideal estimation performance, more work on polynomial curve fitting is required, such as the order of polynomials chosen.

Third, LPF requires no boundary modifications. Boundary modifications in polynomial curve fitting are a very difficult task.

Introduction to Local Polynomial Fitting

Let X and Y be two random variables whose relationship can be modelled as

$$Y = m(X) + \sigma(X)\epsilon \;\; E\epsilon = 0, \;\; var(\epsilon) = 1, \tag{4.96}$$

where X and ϵ are independent. Of interest is to estimate the regression function $m(x) = E(Y|X = x)$, based on $(X_1, Y_1), ..., (X_n, Y_n)$, a random sample from (X, Y). If x is not a random variable but y is, their relationship can be modelled as

$$Y = m(X) + \epsilon \;\; E(\epsilon) = 0, \;\; var(\epsilon) = 1.$$

If the $(p+1)$th derivative of $m(x)$ at the point x_0 exists, we can approximate $m(x)$ locally by a polynomial of order p:

$$m(x) \approx m(x_0) + m'(x_0)(x - x_0) + ... + m^{(p)}(x_0)(x - x_0)^p/p!, \tag{4.97}$$

for x in a neighborhood of x_0, by using Taylor's expansion. From a statistical modelling viewpoint, Equation (4.79) models $m(x)$ locally by a simple polynomial model. This suggests using a locally weighted polynomial regression

$$J_h(x) = \sum_{i=1}^{n} \{Y_i - \sum_{j=0}^{p} \beta_j (X_i - x_0)^j\}^2 K(\frac{X_i - x_0}{h}), \tag{4.98}$$

where $K(\cdot)$ denotes a non-negative weight (kernel) function and h - bandwidth - determines the size of the neighborhood of x_0. If $\hat{\beta} = (\hat{\beta_0}, ..., \hat{\beta_p})$ denotes the solution to the above weighted least squares problem, then by Equation (4.97), $\hat{\beta}$ estimates $m(x)$.

It is more convenient to write the above least squares problem in matrix notation. Denote by W the diagonal matrix with entries $W_i = K((X_i - x_0)/h)$. Let X be the design matrix whose (l, j)th element is $(X_l - x_0)^{j-1}$ and put $y = (Y_1, ...Y_n)^T$. Then, the weighted least squares problem (4.98) can be written in matrix form as

$$\min_{\beta} (y - X\beta)^T W (y - X\beta),$$

where $\beta = (\beta_0, ...\beta_p)^T$. Ordinary least squares theory provides the solution

$$\hat{\beta} = (X^T W X)^{-1} W y,$$

whose conditional mean and variance are

$$\begin{aligned} E(\hat{\beta}|X_1,...X_n) &= (X^T W X)^{-1} W m \\ &= \beta + (X^T W X)^{-1} W r, \\ var(\hat{\beta}|X_1,...X_n) &= (X^T W X)^{-1}(X^T \Sigma X)(X^T W X)^{-1}, \end{aligned} \tag{4.99}$$

where $m = m(X_1),...m(X_n)^T$, $r = m - X\beta$, the residual of the local polynomial approximation, and $\Sigma = diag[K^2\{(X_i - x_0)/h\}\sigma^2(X_i)]$.

At first glance, the above regression approach looks similar to the traditional parametric approach in which the function is usually *globally* modelled by a polynomial. In order to have a satisfactory modelling bias, the degree M of the polynomial often has to be large. But this large degree M introduces an over-parametrization, resulting in a large variability of the estimated parameters. As a consequence the estimated regression function is numerically unstable. In marked contrast to this parametric approach, the technique is *local*, and hence requires a small degree of the local polynomial, typically of order $p = 1$ or occasionally $p = 3$.

4.3.4 Simulations and Experiments

To illustrate the behavior of using the unlabelled training data method for solving the small sample size problem, which is proposed in the foregoing section, we present here a numerical study, which shows the result that the theory in this chapter works. All of the simulation tasks are completed in MATLAB.

For the simulation study, we want to build a two-input and one-output model. Their relations are as follows.

$$\begin{aligned} x_1(t) &= 0.5 + 0.4\sin(2\pi t) + \epsilon_1 \\ x_2(t) &= -0.6 + 0.3\cos(2\pi t) + \epsilon_2 \\ Y(t) &= \cos(x_1) + \sin(x_2) + x_1 x_2 + \epsilon_3, \end{aligned} \tag{4.100}$$

where, the distributions of ϵ_1, ϵ_2 and ϵ_3 have the same form $N(0, \sigma^2)$ and $\sigma = 0.01$. The sampling rates are identical as $\Delta = 0.1s$ both for training data and testing data. For training data, the time interval is $t_0 = 0$ and $t_T = 1s$, such that total $n_o = 11$ data points in table 4.3 and for testing data, the time interval is $t_0 = 0.05$ and $t_T = 0.95s$, such that total $n_1 = 10$ data points in table 4.4. Since the maxim frequency Ω_h of the system in Equation (4.100) is $1Hz$ and the sampling frequency Ω_s is $10Hz > 2\Omega_h$, it satisfies the "*Shannon*" Theorem.

We use a two-layer radial basis network structure. The first layer has radial-basis neurons. The second layer has pure-line neurons, and we calculate its weighted input with dot-prod. Both layers have biases. We used MATLAB function "*newrbe*" with "spread=0.1", to train the neural network. Then, we use the Equation (4.101) as evaluating function with the testing data for

Table 4.3. The training data sample.

x_1	x_2	Y
0.5104	-0.3080	0.4318
0.7315	-0.3574	0.1323
0.8938	-0.5146	-0.3163
0.8905	-0.7076	-0.6519
0.7373	-0.8340	-0.6232
0.4970	-0.9027	-0.3515
0.2674	-0.8584	-0.0114
0.1177	-0.6967	0.2528
0.1188	-0.5087	0.4345
0.2719	-0.3806	0.4863
0.4920	-0.3136	0.4233

Table 4.4. The testing data sample

x_1	x_2	Y
0.6054	-0.3213	0.3026
0.8347	-0.4217	-0.0964
0.8986	-0.6179	-0.5057
0.8347	-0.7902	-0.7121
0.6292	-0.8833	-0.5097
0.3812	-0.87067	-0.1597
0.1696	-0.7727	0.1501
0.1028	-0.6044	0.3709
0.1631	-0.4242	0.5190
0.3692	-0.3139	0.5111

testing the network performance. Let Y_i be an actual test data output and $\hat{Y}_i$ be the corresponding neural network's output,

$$err = \sqrt{\frac{1}{n_1}\sum_{i=1}^{n_1}(Y_i - \hat{Y}_i)^2}. \tag{4.101}$$

When the origin $n_o = 11$ training data for learning, the result of err_o is 0.1607.

Then, we produced the unlabelled training data with MATLAB function "*spline*" using the origin $n_o = 11$ training data with the scales $k = 2, 3, 4, 5, 6, 7$ and sampling rate $\Delta_k = \Delta/k$. By removing the end tail much noisy data, we produced six groups of unlabelled data, $(n_k = k(n_o - 1))$. Then, using the unlabelled training data to train the neural networks individually, the results of err_k are shown in Table 4.5.

Table 4.5. The results of learning error with unlabelled training data.

k	n_k	err_k
2	20	0.0486
3	30	0.0461
4	40	0.0408
5	50	0.0316
6	60	0.0262
7	70	0.0267
9	90	0.1059

As analysis in Section 4.3.1, with regard to Equation (4.91), when n_k increases, the mean-square error of learning decreases. However, after the n_k reached a specific lever, here $n_k = 70$, the second term in Equation (4.91) did

dominate the total error, but not the first and third terms. Thus, the error increased.

Experimental Study

The aim of this experiment is to illustrate how to use the proposed local polynomial fitting interpolation re-sampling approach and verify it.

The control problem involves tracking Gyrover in a circular path with a defined radius. A human expert controls Gyrover to track a fixed 1.5-meter radius circular path and collect around 1,000 training samples with a sampling rate of $0.062s$ ($62ms$). Table 4.6 displays some raw sensor data from the human expert control process.

Table 4.6. Sample human control data.

Input				Output
β	$\dot{\beta}$	β_a	$\dot{\alpha}$	U_1
5.5034	0.9790	2.4023	0.8765	179.0000
5.7185	1.2744	2.3657	1.4478	176.0000
5.6012	-0.8374	2.1313	1.0767	170.0000
5.1271	0.6641	2.1460	0.6030	170.0000
5.9433	-0.4907	1.0425	1.3574	143.0000

We build up an ANN learning controller based on learning imparted from expert human demonstrations. Here, the ANN is a cascade neural network architecture with node-decoupled extended Kalman filtering learner (CNN), which combines (1) flexible cascade neural networks, which dynamically adjust the size of the hidden neurons as part of the learning process, and (2) node-decoupled extended Kalman Filtering (NDEKF), a faster converging alternative to quickprop algorithms. This methodology has been proven to efficiently model human control skills [76].

Achieving this goal requires two major control inputs: U_0 controlling the rolling speed of the single wheel $\dot{\gamma}$, and U_1 controlling the angular position of the flywheel β_a. For the manual-mode (i.e., controlled by a human), U_0 and U_1 are input by joysticks, and in auto-mode they are derived from the software controller. During all experiments, we fix the value of U_0.

Based on these original training sample data, the CNN learner exhibited poor learning performance, and Gyrover always fell down soon after an initial running. Using this original learning controller, Gyrover is unable to track a circular path. Subsequently, the human expert controlled Gyrover to track the circular path two times and produced around 17,000 and 18,000 training samples respectively. By combining these two sets of data, the performance of the CNN learner was more effective and the experiment was successful. Thus,

the reason for the failure initially was due to the limited number of training samples.

In this work, we used the approach outlined above to solve the small sample problem relying on the first set of original training sample data. First, we carried out the interpolation for lean angle β with regard to time t using a local polynomial fitting algorithm. In MATLAB, we used "*lpolyreg*" to collect the polynomial coefficients and the bandwidth and then used "*lpolyregh*" to calculate the unlabelled training data, $\acute{\beta}$. Figure 4.18 shows the local polynomial fitting for lean angle β.

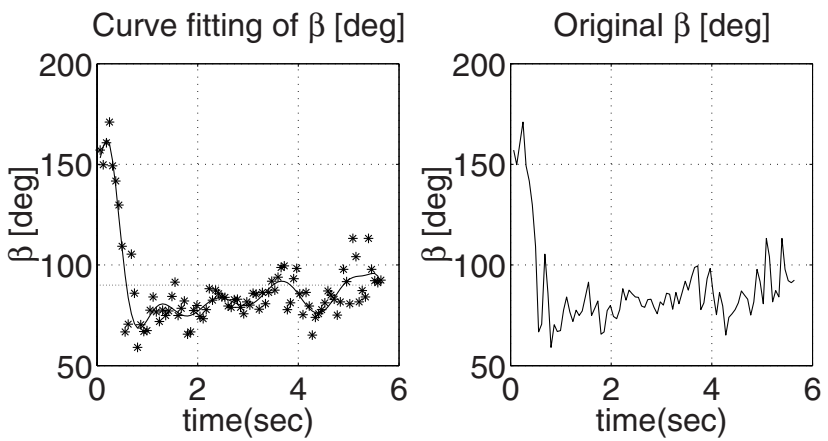


Fig. 4.18. Local polynomial fitting for lean angle β.

Second, we performed the function estimation for $(\dot{\beta},\ \beta_a,\ \dot{\alpha},\ U_1)$ as for lean angle β. We produced the new unlabelled training sample by interpolation with the new sampling rate of $0.0062s$ $(6.2ms)$ and produced about 10,000 pieces of unlabelled training sample data.

Third, by putting these 10,000 pieces of unlabelled sample data into a new CNN learning model, we developed a new CNN-new neural network model. We can, therefore, compare it with the old CNN model, CNN-old, which is produced by the original single-set training data. We use β and β_a of another set of human demonstration training data as inputs to compare the outputs of these two neural network models. Figure 4.19 shows the comparison among the human control, the CNN-new model learning control and the CNN-old model learning control. As Figure 4.19 demonstrates, the CNN-new model has smaller error variance than that of the CNN-old, i.e., it may have better learning control performance. Practical experiments confirm this.

Subsequently, we use the CNN-new model as a learning controller to let Gyrover track a circle automatically in real-time. This experiment was also

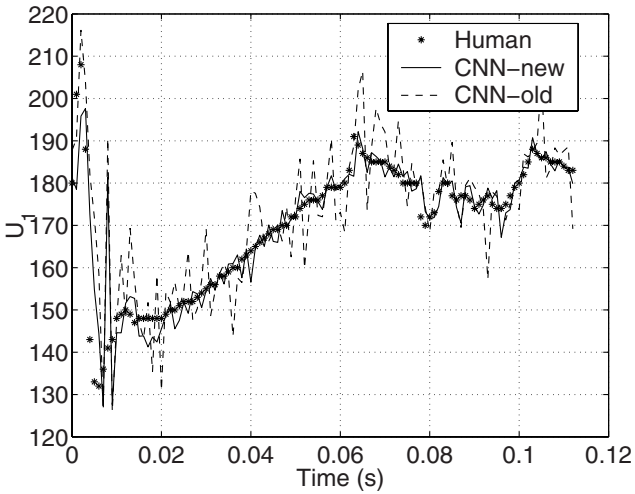


Fig. 4.19. Comparison of U_1 in a set of testing data.

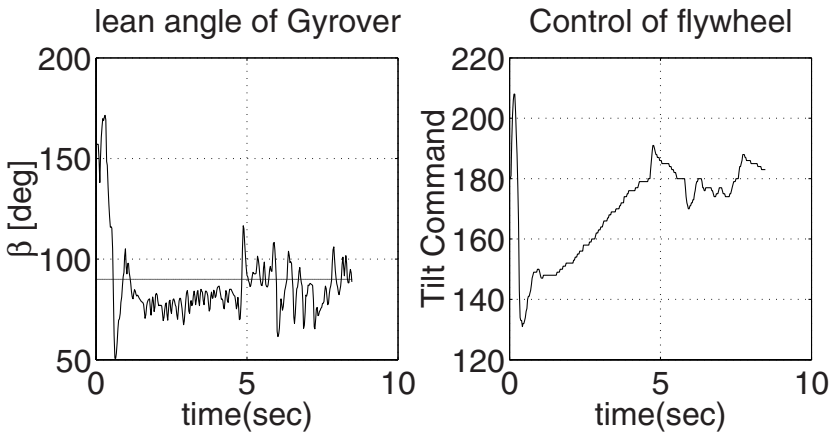


Fig. 4.20. Human control.

successful. Figures 4.20 and 4.21 show the control state variables lean angle β and control command U_1 for a set of human and learning controls.

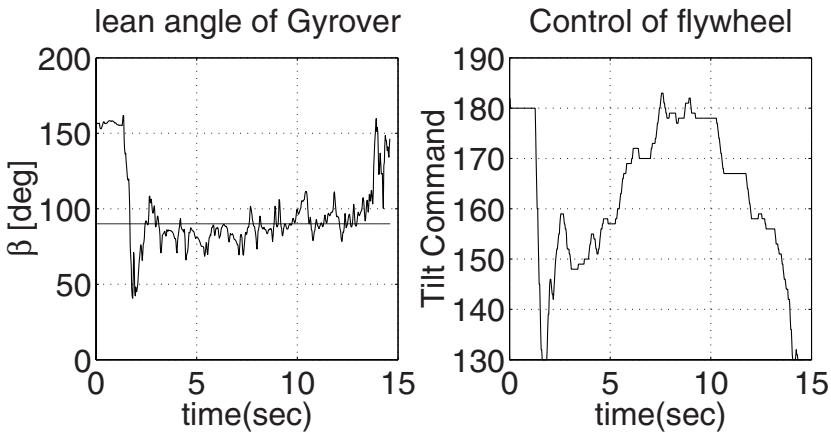


Fig. 4.21. The CNN-new model learning control.

5

Further Topics on Learning-based Control

5.1 Input Selection for Learning Human Control Strategy

Modeling human control strategy (HCS) refers to a model of a human expert's control action in response to system real-time feedback. That is, we aim to develop a relationship between human control action (control commands) and the system response (state variables).

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

$$\Phi : I \longrightarrow O, \tag{5.1}$$

where I is the model input and O is the model output. Modeling HCS means that the function Φ is derived by empirical input-output data. The major input data are state variables and the output data are control variables. In order to generate reliable models, we should not neglect some important factors, such as previous control inputs. However, if we consider all of these as model inputs, the dimension of the input space is too large and there is an excess of highly redundant information. In fact, it is important that not all features presented are taken into account. The following three types of undesirable features should be discarded or neglected:

1. irrelevant features that do not affect the output in any way;
2. redundant features that are linearly related to or dependent on other features;
3. weakly relevant features with only trivial influences on the model output.

Here, the input selection for this class of regression problems involves the removal of irrelevant features, marginally relevant features and redundant features, and selecting a group of the smallest self-contained subset of features from the full set of features with respect to the learning output. This feature selection is a process independent of any special neural network architecture and the training process. Moreover, we assume that the full features already

Y. Xu and Y. Ou: Control of Single Wheel Robots, STAR 20, pp. 119–149, 2005.

include all the necessary information to determine the learning output. It means that the input selection process will not produce any new feature.

When using random sampling, or experimental designs, several factors can be varied independently. Factors can be varied one-at-a-time (OAT) and all the other factors can be held constant, at the same time. Modeling human control strategy is not an OAT process. All the effective factors (system states or parameters) are nonlinearly relevant to each other, i.e. if one of these factors changes, then all the other key parameters will also change.

Different feature selection methods have been analyzed in the past. Based on information theory, Battiti [9] proposed the application of *mutual information* critera to evaluate a set of candidate features and to select an informative subset to be used as input features for a neural network classifier. In [54], interclass and intraclass distances are used as the criterion for feature selection. A method proposed by Thawonmas [105] performs an analysis of fuzzy regions. All these approaches are suitable for classification problems. Here, modeling human control strategy is a problem of regression. A large set of methods for input selection are based on the analysis of a trained multilayer feedforward neural net work or other specific network architectures in [17], [25], [69] and [101]. Even though, in these methods, the subset features are optimal for their special network architectures, it is possible they are not suitable or optimal for some other network architectures. Our method, while producing similar results in test cases, is applied before learning starts and therefore does not depend on the learning process. In [85] different feature evaluation methods are compared. In particular, the method based on *principal component analysis* (PCA) evaluates features according to the projection of the largest eigenvector of the correlation matrix on the initial dimensions. A major weakness of these methods is that they are not invariant under a transformation of the variables. Some nonlinear PCA approaches are examined in the literature [50]-[51]. In fact, the process of building up a nonlinear PCA is similar to the training process for a feedforward neural network. Another large class of methods are called *filter*, *wrapper* and a combination of them [19], [56] and [68]. These methods are focused on a searching strategy for the best subset features. The selection criteria are similar to the above approaches. The searching strategy is very import for classification problems with hundreds or thousands of features. However, for regression problems there are usually fewer features.

Feature selection is one of the initial steps in the learning process. Therefore, we propose to analyze through some relatively simple and efficient processes, the search for the smallest subset of full features. This process of *input selection* ensures that learned models are not over-parameterized and are able to be generalized beyond the narrow training data. Our approach can be divided into three subtasks:

1. significance analysis for the full features;
2. dependence analysis among a select set of features;
3. self-contained analysis between the select set of features and the learning output.

5.1.1 Sample Data Selection and Regrouping

Our learning control problem can be considered as building a map Φ between the system states X and the control inputs Y. We assume both $X = [x_1, x_2, ..., x_m]$ and Y may be time-variant vectors with continuous second order derivatives. Furthermore, without the loss of generality, we restrict Y to be scalar for the purposes of simplifying the discussion.

$$\Phi : Y = f(X) = f(x^1, x^2, ..., x^m). \tag{5.2}$$

Here, we focus our attention on the estimation of regression function f. For a fairly broad class of sampling schemes t_i, they have fixed and identical sampling time intervals $\delta t > 0$ ($\delta t = t_{i+1} - t_i$). We have a sample data table, which includes the observation for both inputs and outputs to train a neural network as a learning controller. The data table is derived from a discrete time sampling from human expert control demonstrations. A training data point $T_j = [X_j^o, Y_j^o]$ ($j = 1, 2, ..., N$) consists of a system state vector X_j^o and the control input Y_j^o. Let (X^o, Y^o) be a data point from the sample data table and Y^o be the observations for true Y and

$$Y^o = Y + \epsilon, \tag{5.3}$$

where ϵ has the Gaussian distribution with $E(\epsilon) = 0$ and variance $\sigma^2 > 0$. If X^o is the observation of X,

$$X^o = X + \varepsilon, \tag{5.4}$$

where $\varepsilon \in R^m$. Here, we assume each system state x_i has a similar observation error as Y. The distribution of ε has the form $N(0, \sigma^2 \mathbf{I}_m)$. X form the full feature set. We will select the key features from them with respect to the learning output (or control input) Y. For the input selection, we need to carefully select and collect the sample data into a number of new local groups. The sample data table may be produced by one or a combination of several human control processes. In the (m) dimension data space, according to the distance D, we cluster the data into a small ball with radius r ($D < r$), as shown in Figure 5.1

$$D = ||X_i - X_c|| = \sqrt{(x_i^1 - x_c^1)^2 + (x_i^2 - x_c^2)^2 + ... + (x_i^m - x_c^m)^2}, \tag{5.5}$$

where X_c is the center data point of the ball (or data group).

In any given ball, even though the values of the data are very similar, the first-order time derivatives of them may differ sufficiently. This is because the near data points may come from different control processes or different paragraphs in the same control process. The selection criteria for a data point T_j are as follows.

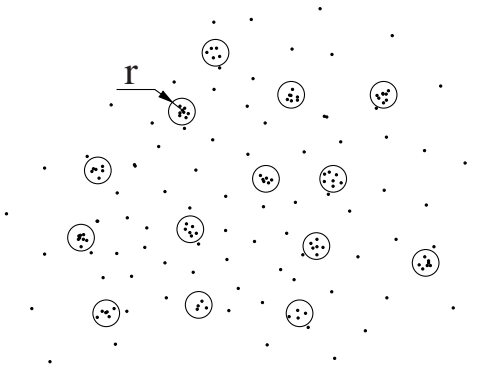


Fig. 5.1. Clustering the data into a small ball with radius r.

1. If we define η as the signal to noise ratio and for any select data point $X_j = [x_j^1, x_j^2, ...x_j^m]$,
$$\delta x_j^i = x^i(j+1) - x^i(j), \tag{5.6}$$
$$|\frac{\delta x_j^i}{\delta t}| > \eta\sigma/\delta t, \quad \forall\, i = 1, 2, ..., m, \tag{5.7}$$
where $\eta = 1$ for significance analysis and $\eta = 10$ for dependent analysis and σ is the variance of noise defined in Equation (5.3).
2. For each data group (or ball), let $\bar{N}_k$ be the number of the k^{th} data group, $\bar{N}_k$ ought to be larger than $\bar{N}$, where
$$\bar{N} = max\{20, 2m\}. \tag{5.8}$$
3. The radius r of the balls may be defined as
$$r = \frac{\sqrt{m}}{\sqrt{\eta}\xi}, \tag{5.9}$$
where
$$\xi = max\{|\frac{\partial^2 f}{\partial x_i, \partial x_j} / \frac{\partial f}{\partial x_i}|\} \quad \forall\, i, j = 1, 2, ..., m.$$

Next, the regrouping strategy is as follows.

1. Remove the data points that do not satisfy condition 1 and reorder the data points.
2. Select the first data point T_1 as the center of the first data group.
3. Calculate the distance D_1 between the next data point with the center of the first data group. If $D_1 \leq r$, the data point is set to the first data group.
4. If $D_1 > r$, calculate the distance D_2 between this data point with the center of the second data group. If $D_2 \leq r$, the data point is set to the second data group. If $D_2 > r$, this data point will be tested with the third

group. If this data point can not be set into any group, it will be set as the center of a new data group.
5. Repeat the last step, until all data points have been set into a group.
6. Remove the data groups where the number of data points is less than $\bar{N}$ and reorder the group number.

Here, we assume the data is large enough in the sense of this regrouping process. Finally, we obtain k balls (or k groups) of sample data, which satisfy the above conditions.

5.1.2 Significance Analysis

We define $I(x_i)$ as the significance value of x_i in respect to Y, given that $i = 1, 2, ..., m$ and if $I(x_i) > I(x_j)$, x_i is more important than x_j.

Here, for the significance analysis, we mainly focus on significance problems: i.e., what is the important order of the model parameters $x_i, i = 1, 2, ...m$ in respect to Y.

Local Sensitivity Analysis

Local sensitivity analysis (Local SA) concentrates on the local impact of the factors on the model. Local SA is usually carried out by computing partial derivatives numerically. Local SA is especially suitable for human learning control strategy, because the time interval of data sampling is the same for all the variables. Our global significant order is derived from the cumulation effect of the local sensitivity analysis process. The sensitivity coefficient $\frac{\partial Y}{\partial x_i}$ is an evaluation of how parameter x_i affects the variable Y. In the followings steps, we will address how to estimate this.

First, with regard to dealing with the local SA, since in each group the data are very near to each other, we transform our nonlinear mapping Equation (5.2) into an almost linear one through a first-order Taylor series expansion:

$$dY = \frac{\partial f}{\partial x_1}dx_1 + \frac{\partial f}{\partial x_2}dx_2 + ... + \frac{\partial f}{\partial x_m}dx_m + o(||dX||). \tag{5.10}$$

X is not the same in each data group, but the difference is very small. Let's consider the effect of this on the almost linear relation. Let $\bar{X}$ be the center data point of this data group. The first-order Taylor series expansion of the coefficient of dx_i is

$$\begin{aligned} \frac{\partial f}{\partial x_i} = \quad & \frac{\partial f}{\partial x_i}\Big|_{X=\bar{X}} + \frac{\partial^2 f}{\partial x_1 \partial x_i}\Big|_{X=\bar{X}}(x_1 - \bar{x}_1) + ... \\ & + \frac{\partial f}{\partial x_m \partial x_i}\Big|_{X=\bar{X}}(x_m - \bar{x}_m) + o(||(X - \bar{X}||). \end{aligned} \tag{5.11}$$

Set

$$A = [a_1, a_2, ..., a_m],$$
$$B = [b_1, b_2, ..., b_m],$$

where

$$a_j = \frac{\partial^2 f}{\partial x_j \partial x_i}, \quad \forall\, i = 1, 2, ..., m,$$
$$b_j = x_j - \bar{x}_j, \quad \forall\, i = 1, 2, ..., m.$$

Then, Equation (5.11) evolves to be

$$\frac{\partial f}{\partial x_i} = \frac{\partial f}{\partial x_i}\Big|_{X=\bar{X}} + AB^T + o(||(X - \bar{X}||). \tag{5.12}$$

$$\begin{aligned} AB^T &\leq \sqrt{(a_1 + a_2 + ... + a_m)^2 (b_1 + b_2 + ... + b_m)^2} \\ &\leq \sqrt{m^2 (a_1^2 + a_2^2 + ... + a_m^2)(b_1^2 + b_2^2 + ... + b_m^2)} \\ &\leq \sqrt{m \xi^2 r^2 (\tfrac{\partial f}{\partial x_i})^2}. \end{aligned}$$

Submitting Equation (5.9) into it we have

$$AB^T \leq |\frac{\partial f}{\partial x_i}|/\eta.$$

Thus, AB^T is a higher order smaller and the effect on the linear relation is small.

We assume that the sampling time interval is small enough. Then, we use finite-difference to approximate the first order differentials.

$$dY \approx \delta Y, \;\; dx_i \approx \delta x_i, \;\; i = 1, ..., m,$$

where $\delta Y = Y(t+1) - Y(t)$ and $\delta x_i = x_i(t+1) - x_i(t)$.

Equation (5.10) will be

$$\delta Y = \frac{\partial f}{\partial x_1}\Big|_{X=\bar{X}} \delta x_1 + \frac{\partial f}{\partial x_2}\Big|_{X=\bar{X}} \delta x_2 + ... + \frac{\partial f}{\partial x_m}\Big|_{X=\bar{X}} \delta x_m + o(||\delta X||). \tag{5.13}$$

In fact, the sample data are the observation for the true values of the variables. From the sample data selection condition 1 in Equations (5.7), we know that the sensor reading error is a higher order smaller. By combining the approximation error and sensor reading error together, we have

$$\delta Y = \frac{\partial f}{\partial x_1}\Big|_{X=\bar{X}} \delta x_1 + \frac{\partial f}{\partial x_2}\Big|_{X=\bar{X}} \delta x_2 + ... + \frac{\partial f}{\partial x_m}\Big|_{X=\bar{X}} \delta x_m + \epsilon', \tag{5.14}$$

where ϵ' partially comes from ϵ, and assums independent Gaussian distribution with $E(\epsilon') = 0$ and $V(\epsilon') = \sigma'^2$.

Second, we use Least-squares theory to estimate $\frac{\partial f}{\partial X}$ in all k data groups. If we let $\frac{\partial f}{\partial X}$ be β, Equation (5.14) becomes

$$\delta Y = \beta_1 \delta x_1 + \beta_2 \delta x_2 + ... + \beta_m \delta x_m + \epsilon', \tag{5.15}$$

where $\beta = (\beta_1, \beta_2, ..., \beta_m)$.

If we let $\bar{N}_i$ be the number of sample data in group i, $(i = 1, 2, ..., k)$. For each group i, we form a $\bar{N}_i \times m$ data matrix X and a $\bar{N}_i \times 1$ data vector Y. Data matrices and vectors are written in roman letters and variables in italic. The least squares estimator $\hat{\beta}$ of β is obtained by minimizing

$$LSE = (\mathrm{Y} - \mathrm{X}\beta)^T(\mathrm{Y} - \mathrm{X}\beta). \tag{5.16}$$

If we take the derivative of Equation (5.16) with respect to β, it yields the normal equation

$$(\mathrm{X}^T\mathrm{X})\beta = \mathrm{X}^T\mathrm{Y}. \tag{5.17}$$

As addressed above, even though X are similar to each other, δX are sufficiently different to each other. Although in most cases, $(\mathrm{X}^T\mathrm{X})^{-1}$ exists, here we do not discuss it, however we will discuss it later. Assuming that it exists, the linear equation (5.17) has a unique solution. Then, pre-multiplying Equation (5.17) by $(\mathrm{X}^T\mathrm{X})^{-1}$ gives the least squares estimator of β, that is,

$$\hat{\beta} = (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{Y}. \tag{5.18}$$

If the above assumption holds, the least squares theory provides us with the following well-known results [38]:

The $k \times 1$ vector $\hat{\beta}$ has the following properties:

1. $$E(\hat{\beta}) = \beta;$$
   that is, $\hat{\beta}$ is an unbiased estimator for β.
2. $\hat{\beta}$ is the best linear unbiased estimator (BLUE) for β, that is, among the class of linear unbiased estimators, $\hat{\beta}$ has the smallest variance. The variance of $\hat{\beta}$ is
   $$Var(\hat{\beta}) = \sigma'^2(\mathrm{X}^T\mathrm{X})^{-1}.$$

As the above Equation shows, and we know $\left.\frac{\partial f}{\partial \bar{X}}\right|_{X=\bar{X}} = \beta$, therefore $\hat{\beta}$ is an ideal estimator for $\left.\frac{\partial f}{\partial \bar{X}}\right|_{X=\bar{X}}$.

The sensitivity coefficient $\frac{\partial Y}{\partial x_i}$ is an estimate of the number of units that change in the variable Y as a result of a unit change in the parameter x_i. This also means that the sensitivity result depends on its physical units of variables and parameters, and is meaningful only when the units of the model are known. In general cases, the variables and the parameters each have different physical units, and therefore the sensitivity coefficients cannot be compared with each other. Here, we assume that all data have been well calibrated. If not, they need to be properly scaled according to the physical meaning.

Global Significance Order

The number of noisy sample data pairs is limited in a local time piece and it is not enough to depend on only one or several local normalized sensitivities to determine the global significance order. We need to combine all of the time piece information to determine the significance order.

The most direct way to define the significance order is sorting the summary of the normalized sensitivity coefficients. To avoid the normalized sensitivity coefficients canceling out with positive and negative values during the summation process, we use their absolute values. Then, we define the Direct Significance Evaluation function as

$$I(x_i) = \frac{1}{k}\sum_{j=1}^{k} |S_j(x_i)|, \tag{5.19}$$

where

$$S_j(x_i) = \hat{\beta}_i^j, \quad i = 1, 2, ..., m, \quad j = 1, 2, ..., k.$$

Usually, this definition works well. However, in some cases where outliers are in evidence, some error data points of a parameter are much larger than others and as such they will greatly affect the average value. In these conditions, we have to seek another definition for the significance evaluation function. According to probability theory, the most important parameter has the largest possibility to be the first important parameter. We can redefine the significance evaluation function in the following steps;

1. In each local time piece, we sort the absolute values of normalized sensitivity coefficients.
2. For each parameter x_i, we collect the times for each possible order. We define $n^l(x_i)$ as the number of times that $|S(x_i)|$ is the l^{th} largest, where $l = 1, ..., m$.
3. We define $p^l(x_i)$ as the possibility that $|S(x_i)|$ is the l^{th} largest, where $l = 1, ..., m$.
$$p^l(x_i) = n^l(x_i)/k \times 100\%,$$
where $n^l(x_i) \geq 0$ and
$$\sum_{l=1}^{m} n^l(x_i) = k.$$
4. For each parameter, we define the Statistical Importance Evaluation function as
$$I(x_i) = p^1(x_i) \times 10^{m-1} + p^2(x_i) \times 10^{m-2} + ...p^m(x_i). \tag{5.20}$$

Thus, the input variables selection can be performed after we work out the significance evaluation values for real learning problems.

5.1.3 Dependency Analysis

First, we will use an example to distinguish three terms, "relevant", "dependent" and "(linear) relative", which will be used in this chapter. Suppose that x_1, x_2, x_3 and x_4 have the following relations

$$\begin{cases} x_3 = & x_1 x_2 \\ x_4 = 3x_1 + 2x_2. \end{cases}$$

Then the relation between x_1 and x_3 is "relevant", but not "dependent" or "(linear) relative". The relation among x_1, x_2 and x_3 is "dependent" and "relevant", but not "(linear) relative". The relation between x_1, x_2 and x_4 is "(linear) relative", "dependent" and "relevant". The "dependent" relation is important in the later two subtasks described in Section 5.1.3.

Let $\acute{X} = [\acute{x}_1, \acute{x}_2, ..., \acute{x}_{\bar{m}}]$, be a subset of X and $\bar{m} \leq m$. Let's consider the relation among $\acute{X}$. Supposed that the relationship of them is "dependent" and

$$\acute{x}_{\bar{m}} = f'(\acute{x}_1, \acute{x}_2, ..., \acute{x}_{\bar{m}-1}). \tag{5.21}$$

We need to use the sample data table to verify this relation. After the data selection and regrouping process, we know that each group of data are very near to each other in the $\bar{m}-1$ dimension data space. Then, by a first-order Taylor series expansion, we transform our nonlinear mapping Equation (5.21) into an almost linear one

$$d\acute{x}_{\bar{m}} = \frac{\partial f'}{\partial \acute{x}_1} d\acute{x}_1 + \frac{\partial f'}{\partial \acute{x}_2} d\acute{x}_2 + ... + \frac{\partial f'}{\partial \acute{x}_{\bar{m}-1}} d\acute{x}_{\bar{m}-1} + o(||d\acute{X}||). \tag{5.22}$$

If there is a "dependent" relation among $\acute{X}$, then they will be almost "(linear) relative" among $d\acute{X}$. As noted earlier, even though $\acute{X}$ are similar to each other, $d\acute{X}$ are sufficiently different from each other.

As the similar analysis before we have

$$\delta\acute{x}_{\bar{m}} = \frac{\partial f'}{\partial x_1}\Big|_{X=\bar{X}} \delta\acute{x}_1 + \frac{\partial f'}{\partial x_2}\Big|_{X=\bar{X}} \delta\acute{x}_2 + ... + \frac{\partial f'}{\partial x_{\bar{m}-1}}\Big|_{X=\bar{X}} \delta\acute{x}_{\bar{m}-1} + \bar{\epsilon}, \tag{5.23}$$

where $\bar{\epsilon}$ is similar to ϵ' which is a higher order smaller with respect to $\delta\acute{x}_{\bar{m}}$. If we let $\frac{\partial f'}{\partial X}\Big|_{X=\bar{X}}$ be $\bar{\beta}$, Equation (5.23) becomes

$$\delta\acute{x}_{\bar{m}} = \bar{\beta}_1 \delta\acute{x}_1 + \bar{\beta}_2 \delta\acute{x}_2 + ... + \bar{\beta}_{\bar{m}-1} \delta\acute{x}_{\bar{m}-1} + \bar{\epsilon}, \tag{5.24}$$

where $\bar{\beta} = (\bar{\beta}_1, \bar{\beta}_2, ..., \bar{\beta}_{\bar{m}-1})$.

We use Least-squares theory to estimate $\bar{\beta}$ in all k data groups in the following Equation.

$$\hat{\bar{\beta}} = (\bar{\mathrm{X}}^T \bar{\mathrm{X}})^{-1} \bar{\mathrm{X}}^T \bar{\mathrm{Y}}. \tag{5.25}$$

To eliminate the possibility that the terms in Equation (5.24), will cancel each other, let

$$||\delta \acute{x}_{\bar{m}}|| = |\bar{\beta}_1 \delta \acute{x}_1| + |\bar{\beta}_2 \delta \acute{x}_2| + ... + |\bar{\beta}_{\bar{m}-1} \delta \acute{x}_{\bar{m}-1}|. \tag{5.26}$$

We define the "(linear) relative" coefficient as

$$\rho = \frac{\bar{\epsilon}}{||\delta \acute{x}_{\bar{m}}||}. \tag{5.27}$$

For each data point in each data group, we will calculate the "(linear) relative" coefficient. For each group, if the group has $\bar{N}_m$ data points, we define the average "(linear) relative" coefficient as,

$$\bar{\rho} = \sqrt{\sum_{i=1}^{\bar{N}_m} \rho_i^2 / \bar{N}_m}, \tag{5.28}$$

where $\bar{\rho}$ ought to be a small number around 0.1 (≤ 0.2).

According to the information derived from these coefficients, we can determine the "(linear) relative" relation among the first-order derivative of $\acute{X}$ and, further, the nonlinear "dependent" relationship among them.

Furthermore, if the relation among $\acute{X}$ is not dependent, then we can use the above analysis to verify the self-contained relation between Y and $\acute{X}$. If they have a self-contained relation, they will have a dependent relation and their first-order derivatives will exhibit a nearly "(linear) relative" relation. It is possible that some other features will affect Y. However, since the "(linear) relative" coefficient is small, the relevance of features should be weak, which means they will be removed from the key feature set.

Thus, we can summarize our input selection approach by the following steps.

1. Perform the significance analysis between the full features and the learning output. Reorder the features according to their significance.
2. Select the first important feature with learning output to carry out the self-contained analysis. If the learning output is dependent on the first significant feature, stop and report the result.
3. If not, combine the first significant feature with the second significant feature to perform a dependent analysis between them. If the second significant feature is dependent on the first significant feature, remove the feature and go to the next step. If the relations are not dependent, execute the self-contained analysis between these two features with the learning output. If the learning output is dependent on these two features, stop and report the result.
4. Combine the features in the former step with the next significant feature to perform a dependent analysis among them. If the next feature is dependent on the former features, remove the feature and go back to the beginning of this step, until there is no feature left, stop and report the result. If the relations are not dependent, execute the self-contained analysis between this new set of features with the learning output. If the learning output

is dependent on these new set of features, go to the next step. If not, go back to the beginning of this step, until there is no feature left, stop and report the result.

5. Remove the features one by one and perform the self-contained analysis to eliminate any redundant features still remaining, until all features are critical. Stop and report the result.

Since some features may be dependent on some later features or the combination of some former features and later features, thus, the last step is necessary and important.

5.1.4 Experimental Study

The aim of this experiment is to illustrate how to use the proposed criterion to realize the input variables selection and validate the approach.

The control problem consists of tracking Gyrover in a circle within a defined radius. In the experimental system, for each sampling process, we can collect 20 sensor readings. Their definitions and physical meanings are shown in Table 5.1. Among them, there are 11 sensor readings, 'ADC0'∼'ADC10', corresponding to the system states variables. There are also two major sensor readings, 'PIR0' (PIC-IN-READ 0) and 'PIR1'(PIC-IN-READ 1), corresponding to the two control inputs: U_0 controlling the rolling speed of the single wheel $\dot{\gamma}$, and U_1 controlling the angular position of the flywheel β_a. For the manual-model (i.e. controlled by a human), U_0 and U_1 are input by joysticks, and in the auto-model, they are derived from the software controller. During all experiments, we only focus on the value of U_1 and fix the value of U_0 to some suitable value. Since the capability of spinning motor for the flywheel is limited, after it drives the flywheel to the working spinning speed $\dot{\gamma}_a$, the motor will try to maintain this speed. Thus, 'PIR3' (PIC-IN-READ 3) can not be a control input and it is fixed during the data producing process, too.

A human expert controlled Gyrover to track a fixed 3-meter radius circular path and produced around $N = 30744$ training samples with identical time intervals ($\delta t = 0.025s$). Table 5.2 displays some raw sensor data from the human expert control process. In the learning control training process, the system states variables are the learning model inputs and tilt command U_1 is the model output. If we put all of the 11 sensor readings, 'ADC0'∼'ADC10', into the learning process, the input dimension is too large to process a reliable model due to the '*curse of dimensionality*'. Therefore we need to perform input variables selection. Here, we use *significance analysis* approach and *dependence analysis* approach for it.

Significance Analysis Study

It is the aim of this step to rank the 11 system variables (model inputs) in significance order with respect to the tilt command (model output).

Table 5.1. Gyrover's sensor data string

Column	Sensor name	Function descriptions
1	Counter read	Spinning angle of flywheel
2	ADC0	Tilt servo position, i.e., tilt angle of flywheel β_a
3	ADC1	Tilt servo current
4	ADC2	Drive motor current
5	ADC3	Roll rate of Gyro, i.e., roll rate of single wheel $\dot{\beta}$
6	ADC4	Pitch rate of Gyro
7	ADC5	Yaw rate of Gyro, i.e., precession rate of single wheel $\dot{\alpha}$
8	ADC6	X accelerometer
9	ADC7	Y accelerometer
10	ADC8	Z accelerometer
11	ADC9	X tilt sensor
12	ADC10	Y tilt sensor, i.e., lean angle of single wheel β
13	PIC-IN-READ 0	Drive command U_0
14	PIC-IN-READ 1	Tilt command U_1
15	PIC-IN-READ 3	Command for spinning speed of flywheel
16	PIC-OUT-READ 0	Drive command
17	PIC-OUT-READ 1	Tilt command
18	PIC-OUT-READ 3	Command for spinning speed of flywheel
19	instantaneous speed	instantaneous driving speed of single wheel $\dot{\gamma}$
20	average speed	instantaneous driving speed of single wheel $\bar{\dot{\gamma}}$ in 5 sampling intervals

Table 5.2. Sample of human control data

Input											Output
ADC0	ADC1	ADC2	ADC3	ADC4	ADC5	ADC6	ADC7	ADC8	ADC9	ADC10	PIR1
2.614	0.029	-0.004	0.422	0.617	1.696	2.440	2.589	3.624	5.821	5.874	180
2.646	1.970	1.989	0.410	0.500	2.104	2.885	2.528	3.178	6.060	6.168	179
2.585	6.245	1.935	-0.300	0.659	2.717	2.724	2.393	3.221	6.021	6.407	178
2.419	3.308	3.475	0.505	0.478	0.310	2.172	2.489	3.214	6.075	5.400	178
2.382	3.352	3.636	-0.678	0.468	1.616	2.601	2.739	3.769	6.075	5.997	175

First, we obtain the sensor reading error information as follows. If we fix Gyrover in some position and start it, all sensor readings should keep in some values. In fact, they are changing around their true values for the noise sake. Then we keep recording the sensor readings for a period of time and collect hundreds of sensor readings for each sensor. From these data, we obtain the noise variance σ for all variables, 'ADC0'∼'ADC10' and 'PIR1' as shown in Table 5.3.

Table 5.3. The noise variance σ information for variables

ADC0	ADC1	ADC2	ADC3	ADC4	ADC5	ADC6	ADC7	ADC8	ADC9	ADC10	PIR1
0.007	0.003	0.006	0.006	0.004	0.004	0.006	0.006	0.007	0.002	0.003	0.019

Second, we select and regroup data points $T_j = [X_j, Y_j](j = 1, 2, ..., 30744)$ from the whole data table as in the following steps.

1. According to Equation (5.6) we calculate the first derivatives for all data points.
2. According to Equation (5.7) we select and place the data points into a new data table with 14,634 data points by removing the data points that their first derivatives are less than the noise ($\eta = 1$).
3. According to Equations (5.8) and (5.9), we work out $\bar{N} = 22$ and $r = 0.5$.
4. According to $r = 0.2$, we cluster the 14634 data points into 6340 groups.
5. According to $N = 22$, we select 45 data groups from the 6340 group by removing the groups in which the data numbers are less than 22.

Third, we perform the local sensitivity analysis (Local SA). According to Equation (5.18), we work out the sensitivity coefficients $\beta = \dfrac{\partial f}{\partial X}$ for all 45 data groups. The local sensitivity coefficients of the first four most significant variables are shown in Figure 5.2.

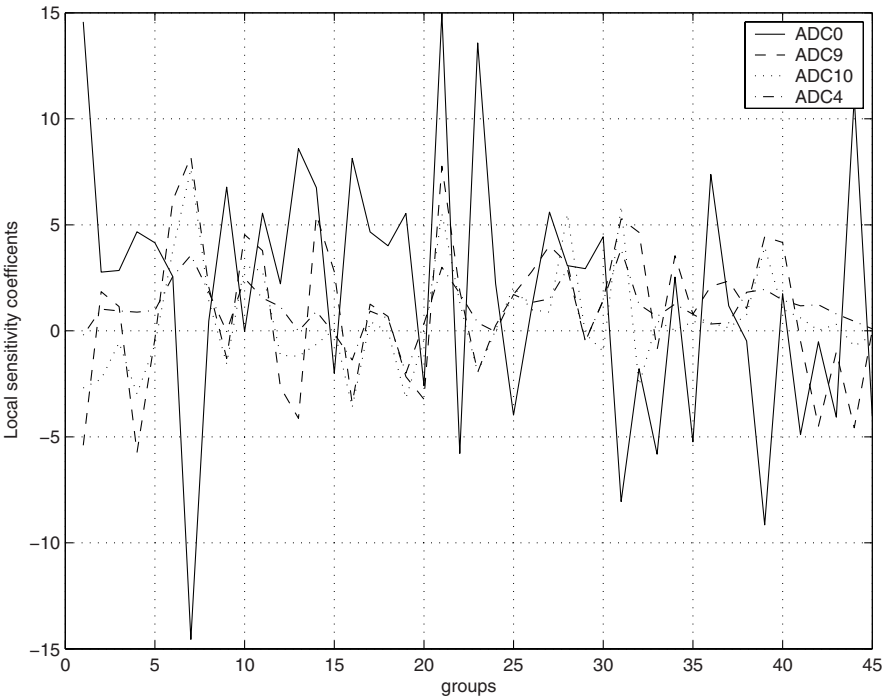


Fig. 5.2. The local sensitivity coefficients of the first four significance variables.

Last, we rank all the system variables in the global significance order. According to the Direct Significance Evaluation function in (5.19), we work out the global significance value $I(x_i)$ for each systems variable and order them in Table 5.3.

Table 5.4. The significance order table

Variables	ADC0	ADC1	ADC2	ADC3	ADC4	ADC5	ADC6	ADC7	ADC8	ADC9	ADC10
$I(x_i)$	331.53	3.07	5.23	25.48	73.99	51.71	15.29	24.91	11.26	173.14	97.89
rank	1	11	10	6	4	5	8	7	9	2	3

We performed the significance analysis experiments several times with different data set combinations. The results are quite similar from time to time.

Dependence Analysis Study

The aim of this step is to obtain the smallest set of system variables and the control input is dependent on this set of features.

First, we execute the self-contained analysis between all of the $m = 11$ system variables with the tilt command.

1. We select and regroup data points $T_j = [X_j, Y_j](j = 1, 2, ..., 30744)$ from the whole data table.
2. According to Equation (5.6), we calculate the first derivatives for all data points.
3. According to Equation (5.7) we select and place the data points into a new data table with 1,085 data points by removing the data points that their first derivatives are less than ten times the noise variance σ ($\eta = 10$).
4. According to r, N, we cluster and select the 1,085 data points into 22 groups.
5. According to Equations (5.24)-(5.28), for each group, we calculate the average "(linear) relative" coefficients that are in Table 5.5.

Table 5.5. The average "(linear) relative" coefficients $\bar{\rho}$ in full system variables

group	1	2	3	4	5	6	7	8	9	10	11
$\bar{\rho}$	0.1011	0.1041	0.1103	0.1252	0.1280	0.1103	0.1232	0.1022	0.1235	0.0903	0.1250
group	12	13	14	15	16	17	18	19	20	21	22
$\bar{\rho}$	0.1257	0.0983	0.1011	0.0865	0.1250	0.1078	0.0990	0.1030	0.1003	0.0316	0.099

Table 5.5 exhibits that the tilt command is dependent on these system variables. However, $m = 11$ is too high to train a learning controller.

Second, according to the significance order in Table 5.4, we add the variable one by one and execute the dependence analysis between the former features with the new features and self-contained analysis between the selected set of features with the tilt command. After the similar process with the full variable self-contained analysis, we stop with a set variables of ('ADC0', 'ADC9', 'ADC10', 'ADC4', 'ADC5', 'ADC3'). Table 5.6 shows part of the average "(linear) relative" coefficients of the self-contained analysis between the

set variables of ('ADC0', 'ADC9', 'ADC10', 'ADC4', 'ADC5') with tilt command. Table 5.7 shows the average "(linear) relative" coefficients of the self-contained analysis between the set variables of ('ADC0', 'ADC9', 'ADC10', 'ADC4', 'ADC5','ADC3') with the tilt command. From Table 5.6, the set features ('ADC0', 'ADC9', 'ADC10', 'ADC4', 'ADC5') can not determine the tilt command and from Table 5.7, the tilt command is almost dependent on the variables of ('ADC0', 'ADC9', 'ADC10', 'ADC4', 'ADC5','ADC3').

Table 5.6. Part of $\bar{\rho}$ in the 5 variables

group	1	2	3	4	5	6	7	8	9	10	11
$\bar{\rho}$	0.5851	0.7654	0.5944	0.6203	0.5336	0.5574	0.5112	0.5799	0.5863	0.6530	0.5334

Table 5.7. $\bar{\rho}$ in the 6 variables

group	1	2	3	4
$\bar{\rho}$	0.1862	0.1654	0.1862	0.1247

Last, we remove the set features of ('ADC0', 'ADC9', 'ADC10', 'ADC4', 'ADC5','ADC3'), one by one to obtain the smallest set of features. The final result is ('ADC0', 'ADC10', 'ADC5', 'ADC3'). The self-contained analysis between the set variables of ('ADC0', 'ADC10', 'ADC5', 'ADC3') with the tilt command is shown in Table 5.8. The table shows that tilt command is almost dependent on the variables of ('ADC0', 'ADC10', 'ADC5', 'ADC3').

Table 5.8. Part of $\bar{\rho}$ in the 4 variables

group	1	2	3	4	5	6	7	8	9	10	11
$\bar{\rho}$	0.1250	0.1198	0.1224	0.1219	0.1168	0.1219	0.1198	0.1250	0.1198	0.1168	0.1219

Learning Control Result

Based on these input variables, we built up an artificial neural network(ANN) learning controller, based on learning imparted from expert human demonstrations. Here, the ANN is a support vector machine learning architecture (SVM), in [81], which can realize a very high learning accuracy with limited data points. We trained an SVM learning controller to control Gyrover. The learning controller works well and the model is very reliable. Figure 5.3 shows the SVM learning control results.

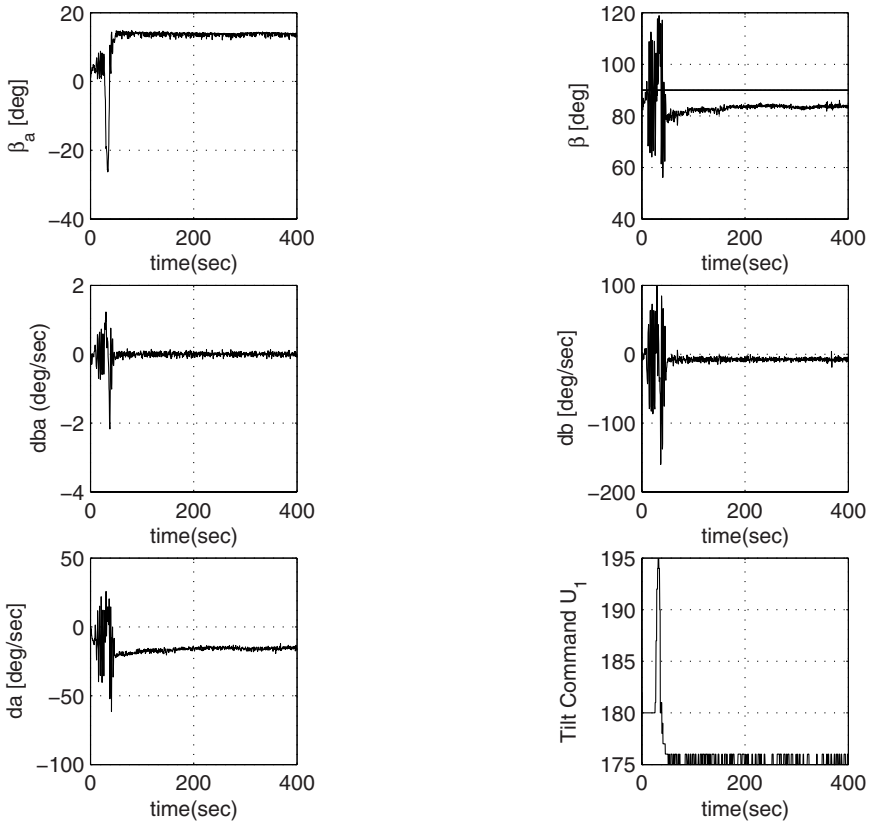


Fig. 5.3. SVM learning control results.

Discussions and Remarks

The most critical and time-consuming part is the data selection and regrouping. The data selection conditions need to be well controlled and parameters need to be carefully selected.

If some of the sensor readings are too noisy to produce the data selection and collection, we may use smoothing technologies. There are many approaches in digital signal processing. An important practical point that needs to be kept in mind is that the state variables in learning human control are usually very low frequency signals.

As described in the significance analysis section, if there are only limited groups, and their $(\mathrm{X}^T\mathrm{X})^{-1}$ do not exist, then remove these groups. If all groups display the problem and it is because there exist two or more variables with strong linear relations, remove one or some of them to avoid the linear relation and perform the significance analysis again.

Moreover, the input variables X may miss some important variables and/or add some almost irrelevant variables. We do not consider the missing problem here, as mentioned in the introduction section: i.e., we assume that we have already included all the important factors. For the latter problems, we divide X into two parts:

$$X = (\acute{X}_1 : \acute{X}_2),$$

where $X \in R^m$, $\acute{X}_1 \in R^{m_1}$, $\acute{X}_2 \in R^{m_2}$ and $m = m_1 + m_2$ and $\acute{X}_2$ is not relevant to Y.

Now suppose that the true model for Equation (5.15) becomes

$$\delta Y = \bar{\beta}_1 \delta X_1 + \epsilon', \tag{5.29}$$

but instead we fit

$$\delta Y = \bar{\beta}_1 \delta X_1 + \bar{\beta}_2 \delta X_2 + \epsilon', \tag{5.30}$$

to the data, where $\bar{\beta}_1 \in R^{m_1}$ and $\bar{\beta}_2 \in R^{m_2}$. We refer to Equation (5.29) and Equation (5.30) as the reduced and full models respectively. Let $\hat{\bar{\beta}}_{1F}$ and $\hat{\bar{\beta}}_{2F}$ be the least squares estimates of $\bar{\beta}_1$ and $\bar{\beta}_2$ in fitting model (5.30) to the data. Let $\hat{\bar{\beta}}_{1R}$ denote the least squares estimates when model (5.29) is fitted to the data. From [90], we have the following two conclusions:

1. $\hat{\bar{\beta}}_{1F}$ and $\hat{\bar{\beta}}_{2F}$ are unbiased estimates of $\bar{\beta}_1$ and $\bar{\beta}_2$, respectively. That is, $E(\hat{\bar{\beta}}_{1F}) = \bar{\beta}_1$ and $E(\hat{\bar{\beta}}_{2F}) = 0$.
2. $E(\hat{\bar{\sigma}}^2_F) = \sigma^2$.

This means in the latter case that it is still an ideal estimate for the sensitivity coefficient $\frac{\partial Y}{\partial x_i}$.

How to choose features is another important aspect that will affect the learning accuracy. Thus, in the next chapter, we will handle this problem.

5.2 Implementation of Learning Control

In this chapter, we show the implementation results of the CNN models trained in the previous chapter. First of all, we validate the CNN models we obtained by applying a Hidden Markov Model based similarity measure. Next, for the experimental implementations of the CNN models, we evaluate the performance of these models by observing the lean angle of the robot and the overall control on the flywheel. Later, we combined the two motions into a single motion. This combined motion ensures that the robot can be fully recovered from the fall position back and balanced at its upright position.

5.2.1 Validation

In this subsection, we will evaluate each of the model generated by the cascade learning algorithm for different behaviors of the robot, including lateral stabilization and tiltup motion. We apply the similarity measure mentioned in Section 3.2.4 to quantify the level of similarity between the original human control data and the model-generated trajectories through simulations. Since we do not have a physical model for these kind of motion for Gyrover, our simulations are done by feeding the current and history state variables and control information into the cascade neural network, to see if it can generate similar control output in each time instant.

Basically, we have two motions to learn: (1) Lateral balancing ($i = 1$), and (2) Tiltup ($i = 2$). For each motion, we give three different set of data for the simulation. For notation convenience, let $X^{i,j}$, $i \in \{1, 2\}$, $j \in \{1, 2, 3\}$, denote the run of different motions i in trail $\#j$.

Vertical Balancing

Figure 5.4, 5.6 and 5.8 show three different vertical balanced motion by human control. The graph on the left of each figure is the plot of lean angle data (β), while the right one plots the orientations of the flywheel (β_a). The corresponding human control data and CNN model control data for $X^{(1,1)}$, $X^{(1,2)}$ and $X^{(1,3)}$ are shown in Figure 5.5, 5.7 and 5.9 respectively. We perform the similarity measure between the human control and CNN model control trajectories for each motion, the results are summarized in Table 5.9. From the performance of this vertical balancing CNN model, we can observe that the model can generate similar control trajectories as human operator, with an average similarity value of 0.5940.

Table 5.9. Similarity measures for vertical balanced control between human and CNN model

	similarity σ
$X^{(1,1)}$	0.5885
$X^{(1,2)}$	0.6235
$X^{(1,3)}$	0.5700
average	0.5940

Tilt-up Motion

Figure 5.10, 5.12 and 5.14 show three different tiltup motion by human control. The corresponding human control data and CNN model control data for $X^{(2,1)}$, $X^{(2,2)}$ and $X^{(2,3)}$ are shown in Figure 5.11, 5.13 and 5.15 respectively.

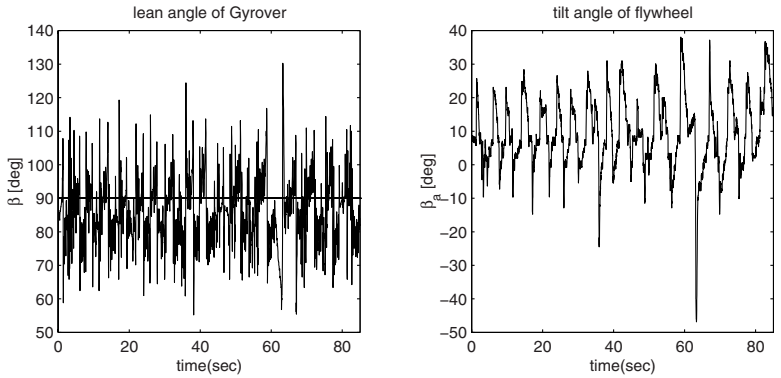


Fig. 5.4. Vertical balanced motion by human control, $X^{(1,1)}$.

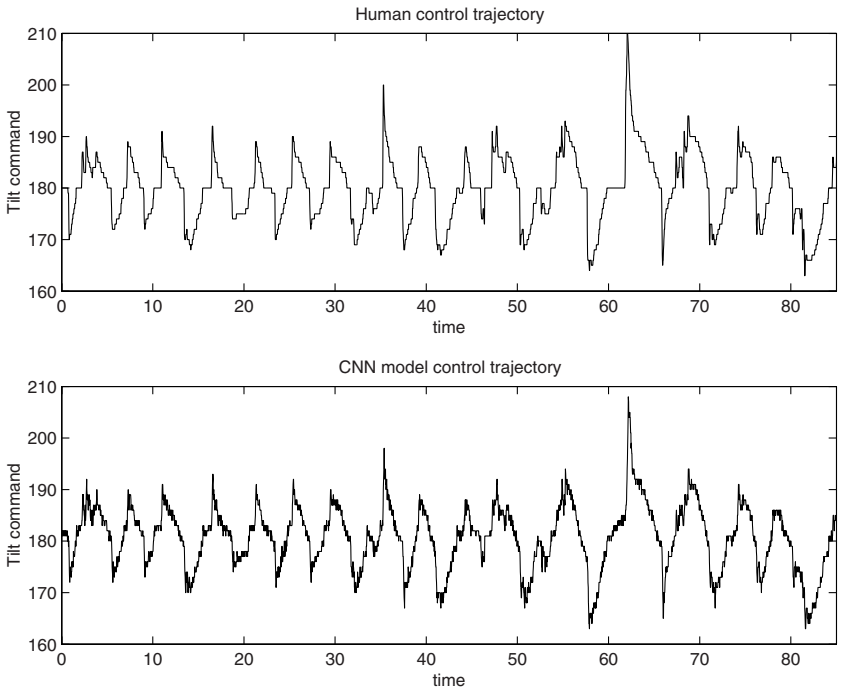


Fig. 5.5. Control trajectories comparison for $X^{(1,1)}$.

Again, we perform the similarity measure between the human control and CNN model control trajectories for each motion, the results are summarized in Table 5.10. The CNN model can also generate similar control trajectories as human operator, with an average similarity value of 0.7437.

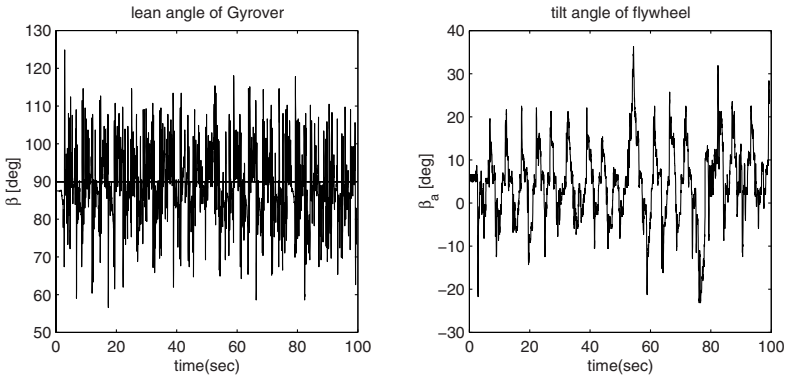


Fig. 5.6. Vertical balanced motion by human control, $X^{(1,2)}$.

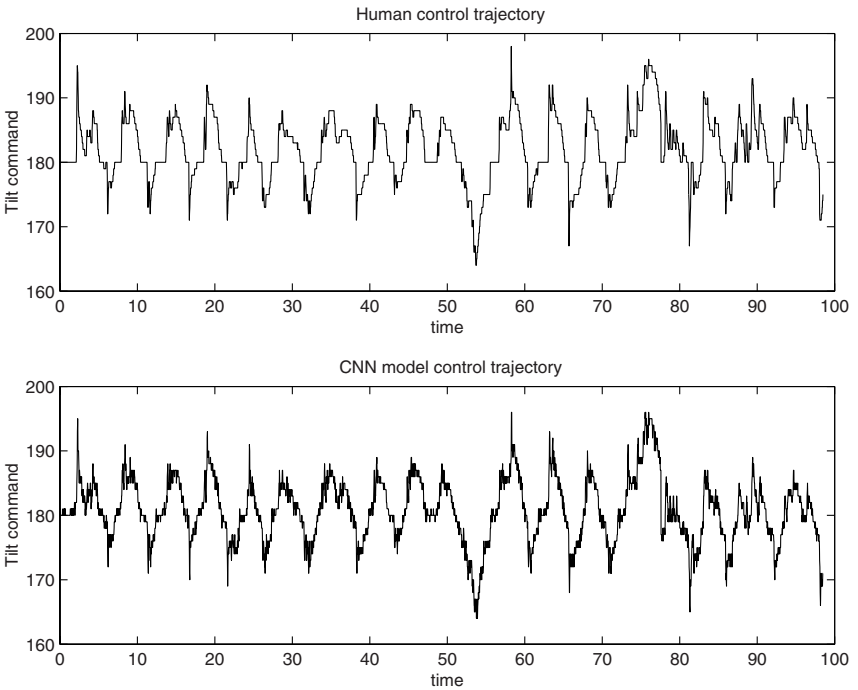


Fig. 5.7. Control trajectories comparison for $X^{(1,2)}$.

Discussions

The simulations we have done in fact is the first step to validate the CNN models we obtained. By using the HMM similarity measure, we compare the human control trajectory with the control trajectory generate from the CNN model of a particular motion. If the similarity measure gives us a relatively

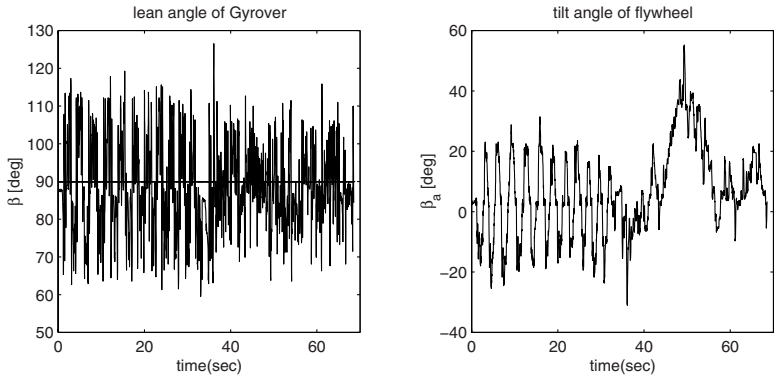


Fig. 5.8. Vertical balanced motion by human control, $X^{(1,3)}$.

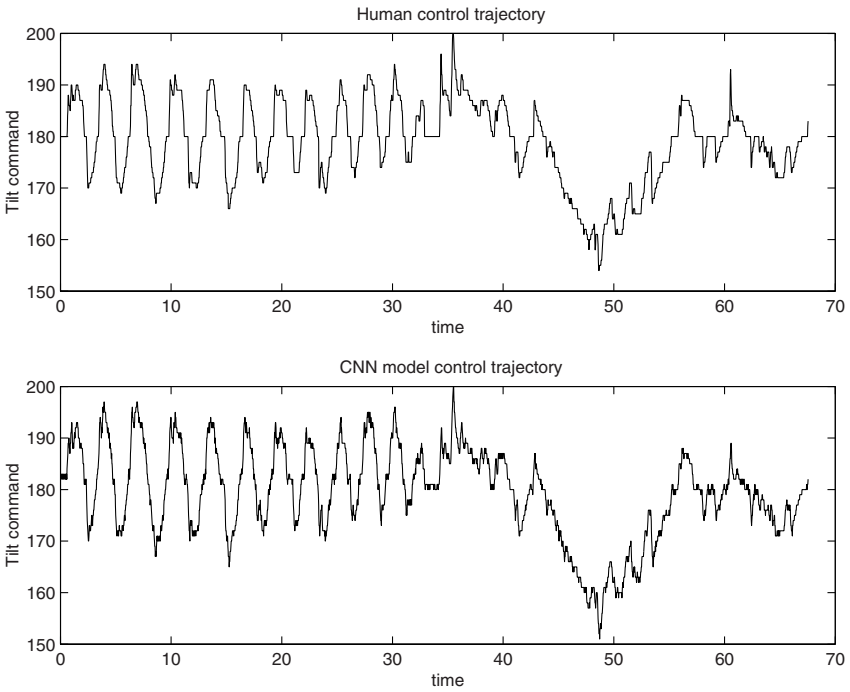


Fig. 5.9. Control trajectories comparison for $X^{(1,3)}$.

high similarity value ($\sigma \geq 0.5$), which implies the particular CNN model can produce 'similar' control output as human control. From the simulation results of the lateral balancing and tiltup motion, we can verify that the CNN models for both motions are able to model the human control strategy. Later on, in the next chapter, we will further verify the models by experimental implementation.

Table 5.10. Similarity measures for tiltup control between human and CNN model

	similarity σ
$X^{(2,1)}$	0.7896
$X^{(2,2)}$	0.7030
$X^{(2,3)}$	0.7386
$average$	0.7437

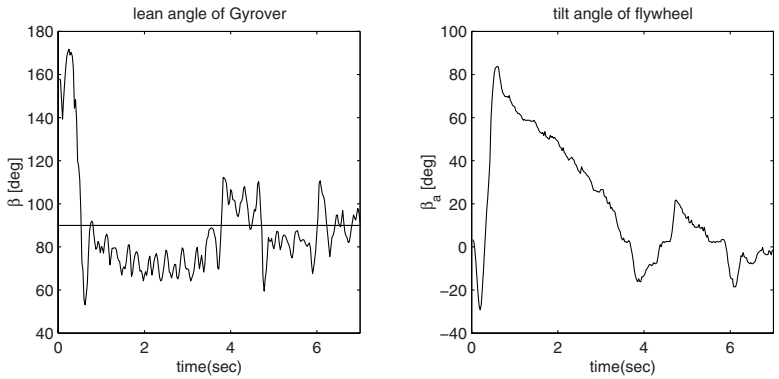


Fig. 5.10. Tiltup motion by human control, $X^{(2,1)}$.

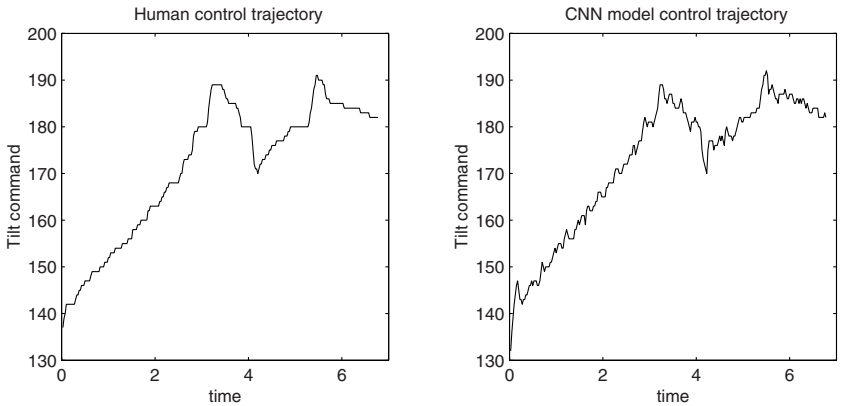


Fig. 5.11. Control trajectories comparison for $X^{(2,1)}$.

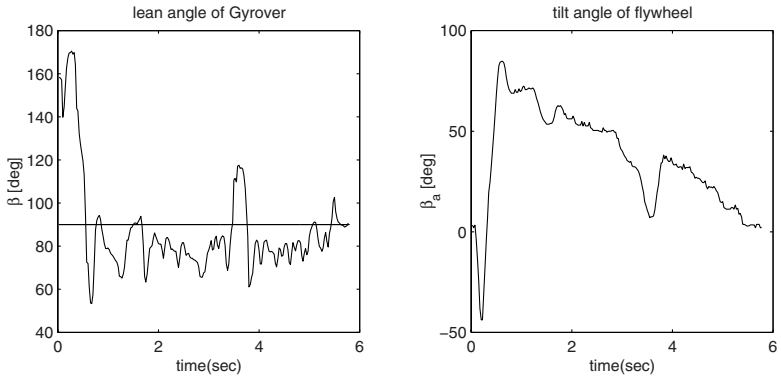


Fig. 5.12. Tiltup motion by human control, $X^{(2,2)}$.

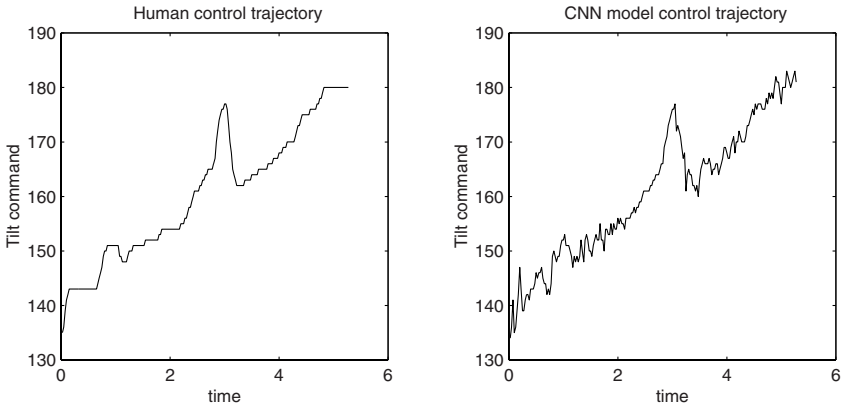


Fig. 5.13. Control trajectories comparison for $X^{(2,2)}$.

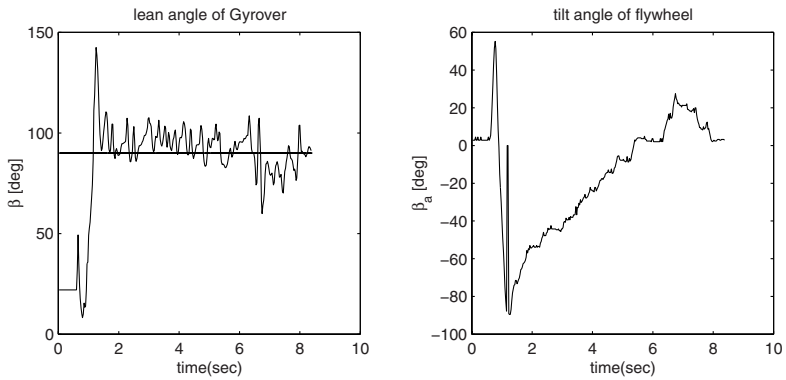


Fig. 5.14. Tiltup motion by human control, $X^{(2,3)}$.

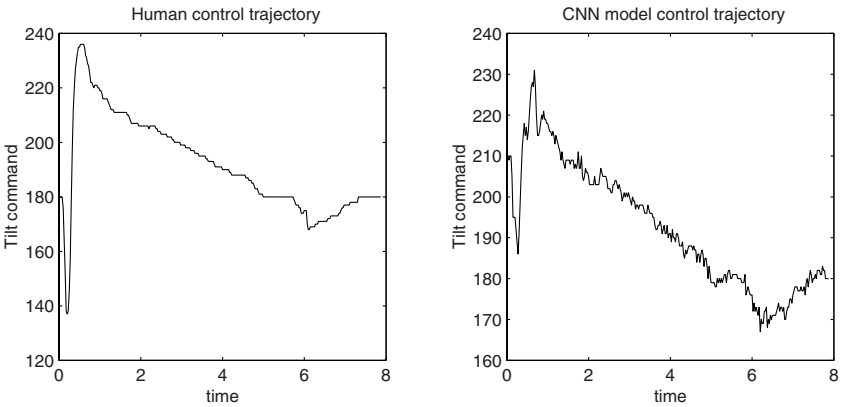


Fig. 5.15. Control trajectories comparison for $X^{(2,3)}$.

5.2.2 Implementation

Vertical Balanced Motion

A number of experiments have been conducted to verify the CNN model for vertical balancing, Figure 5.16, 5.17 and 5.18 shows the implementation results. The human control strategy in balancing the robot at the vertical position is given in Figure 5.19. As mentioned in the pervious chapter, we evaluate the performance by the lean angle of the robot and the degree of freedom remains for the flywheel. We summarized the overall performance of both CNN model and human operator for the vertical stabilized motion in Table 5.11.

Table 5.11. Performance measures for vertical balancing.

	average lean angle $\bar{\beta}$	$DOF_{flywheel}$
CNN control #1	90.24^o	0.9944
CNN control #2	88.11^o	0.8756
CNN control #3	87.57^o	0.8867
Human control	89.41^o	0.9600

When compared with human control, the CNN model we obtained for vertical balancing behaves very similar to human. For the 3 different trails, the CNN model not only able to stablize the robot at around 90^o, but also reserved a high level of degree of freedom for the internal flywheel to oppose any motion that appears to make the robot to fall down.

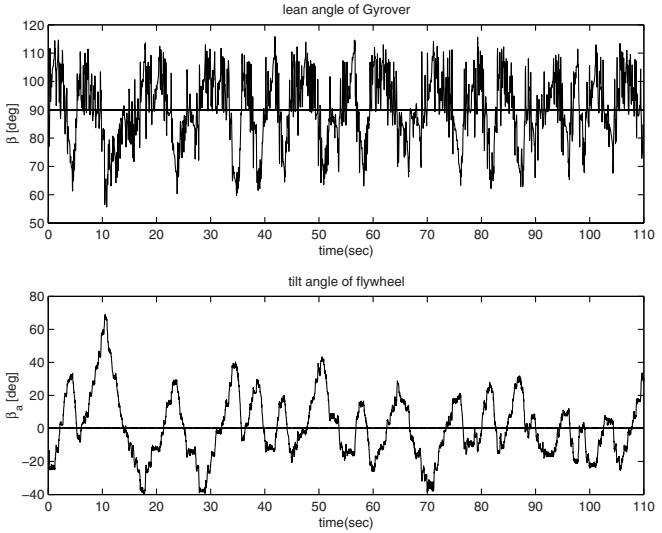


Fig. 5.16. Vertical balancing by CNN model, trail #1.

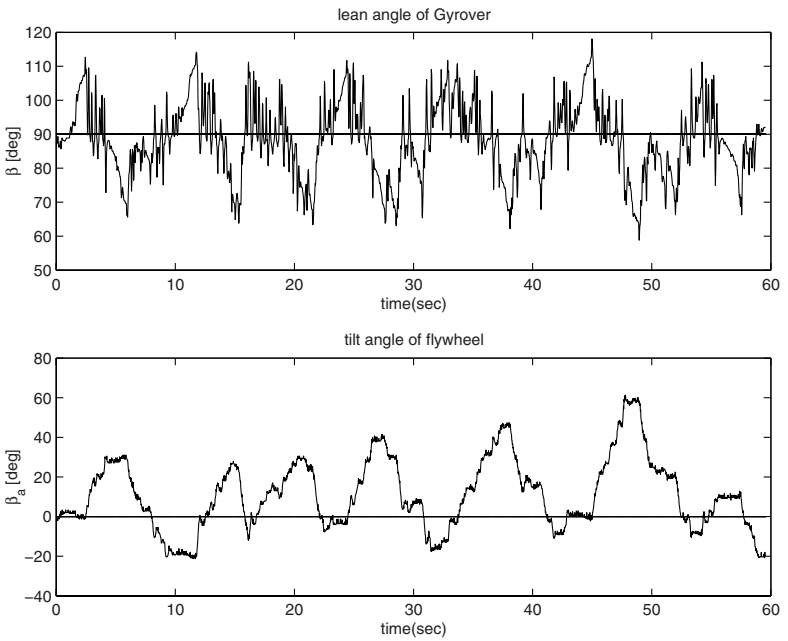


Fig. 5.17. Vertical balancing by CNN model, trail #2.

Tilt-up Motion

Next, we implement another CNN model which is trained by human tiltup motion data, the results are shown in Figure 5.20 and 5.21 for CNN model

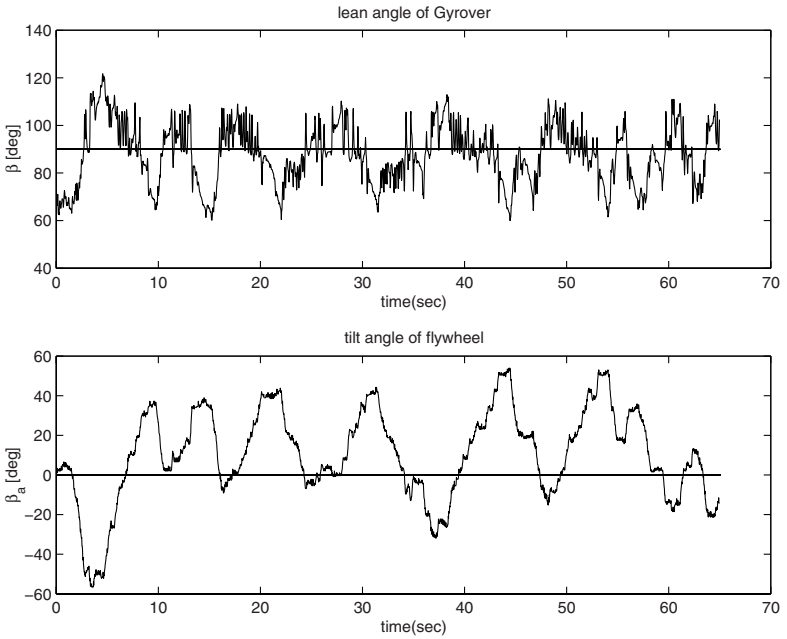


Fig. 5.18. Vertical balancing by CNN model, trail #3.

control, while the human control is shown in Figure 5.22. The performance of these motions are summarized in Table 5.12.

Since a large portion of the flywheel's motion is contributed to tiltup the robot, the overall degree of freedom of the flywheel in tiltup motion is much lower than that of lateral stabilization. For the CNN model control in Figure 5.20 and 5.21, the robot is lying on the ground initially, with $\beta \approx 150^o$, after a few seconds, the model tiltup the robot and brings the robot back to the upright position.

Table 5.12. Performance measures for tiltup motion.

	average lean angle $\bar{\beta}$	$DOF_{flywheel}$
CNN control #1	97.26^o	0.6774
CNN control #2	95.60^o	0.4039
Human control	87.31^o	0.7372

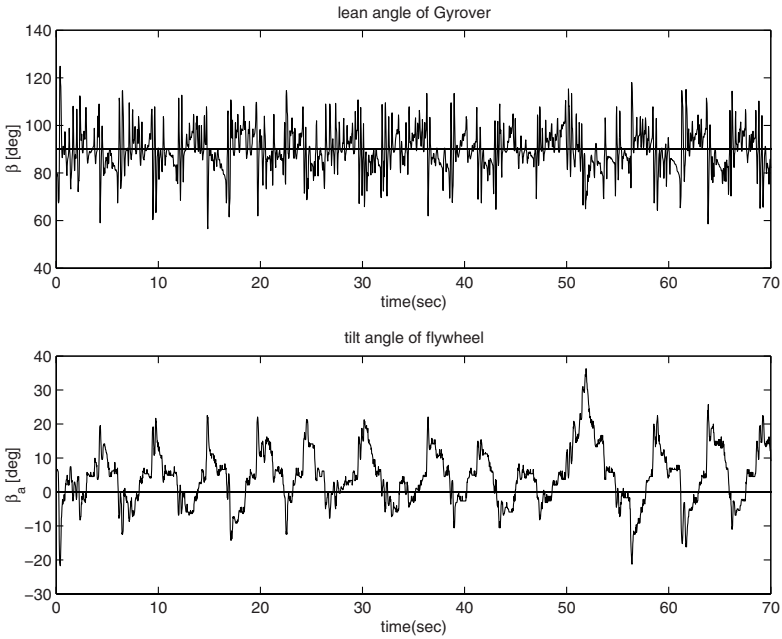


Fig. 5.19. Vertical balancing by human operator.

Combined Motion

We observed that the CNN models for lateral balancing and tiltup motion are subjected to some initial condition, the problem can be solved by combining the two motions to form a single motion.

Consider the case that the robot is in the fall position, that is, with $\beta \approx 150$. In Figure 5.20 and 5.21, although the CNN tiltup model is able to keep the robot to stay around at 90^o for a certain moment, the robot will fall back to the ground eventually because the flywheel has reached an ill-condition ($\beta_a = \pm 90^o$). Moreover, the tiltup model is unable to let the robot to converge to 90^o sometimes, which causes a large fluctuation in the lean angle about 90^o, Figure 5.24.

To deal with this problem, we combine the tiltup motion together with the lateral balanced motion, Figure 5.23. Since the CNN model is unable to keep the robot at the vertical position, after the robot has tiltup, we ask the model to balance the robot at 90^o.

The experimental result for the whole tiltup and stabilization process after the combination is shown in Table 5.13 and Figure 5.25. Initially, the robot is in a fall position, by executing the tiltup control of the CNN model, the robot is recovered to the vertical position. Afterwards, the lateral stabilization is controlled by another model which specifically trained for keeping the robot into the vertical position. From the results, the combined motion can keep

the robot at the vertical position well after tiltup from the ground for a much more longer period of time.

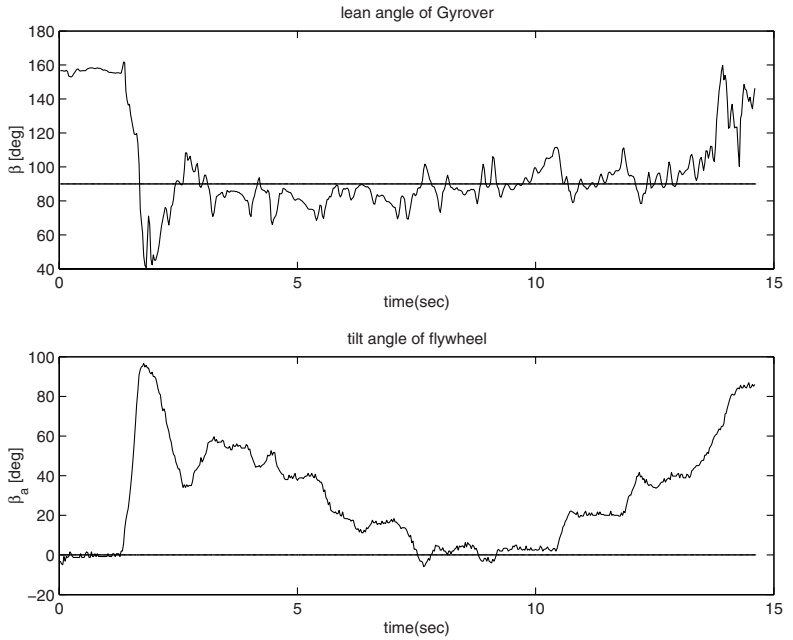


Fig. 5.20. Tiltup motion by CNN model, trail #1.

Table 5.13. Performance measures for combined motion.

	average lean angle $\bar{\beta}$	$DOF_{flywheel}$
CNN control #1	88.40^o	0.8998

5.2.3 Discussions

In this chapter, the CNN models for lateral balancing and tiltup motion are being verified by experimental implementations. By combining the two motions into a single motion, the robot is able to recover from the fall position, and then to remain stable at the vertical position after tiltup. Therefore, we have completed the low-level behavior module within the behavior-based architecture shown in Figure 6.4. With this module completed, we are going to develop a semi-autonomous control for Gyrover in the next chapter.

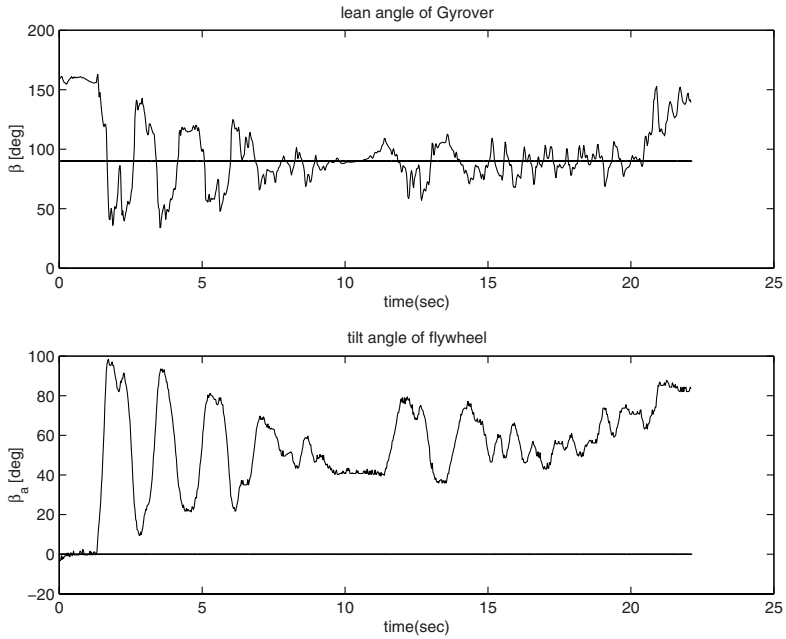


Fig. 5.21. Tiltup motion by CNN model, trail #2.

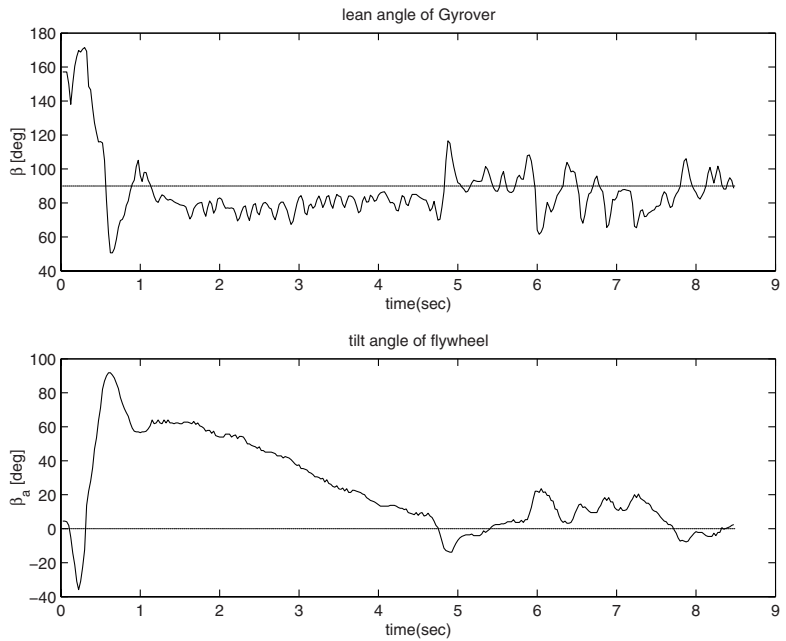


Fig. 5.22. Tiltup motion by human operator.

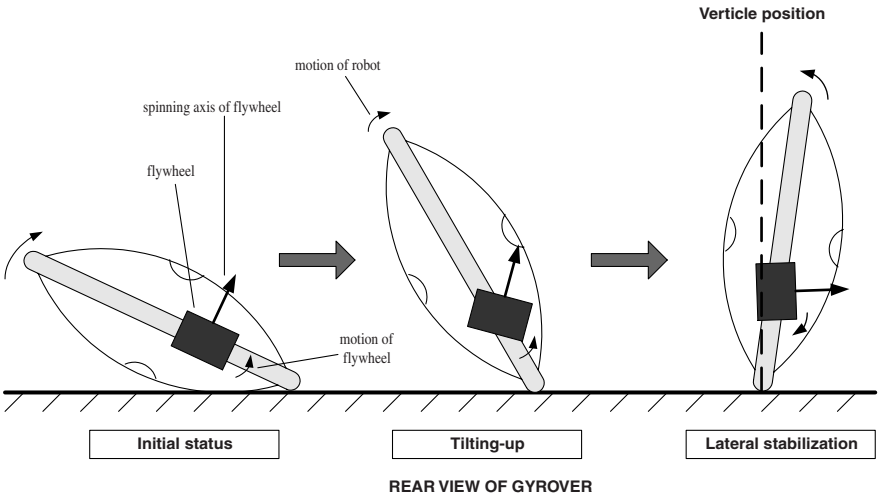


Fig. 5.23. Combined motion.

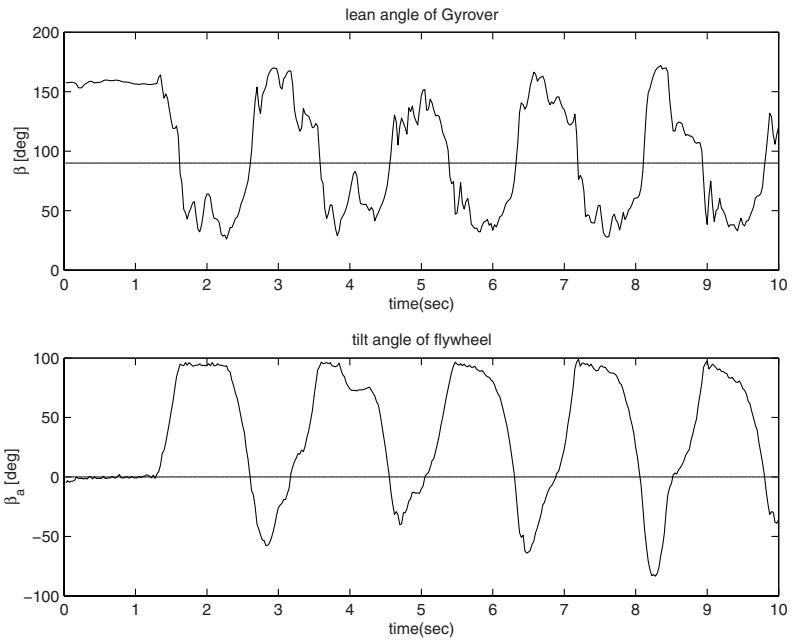


Fig. 5.24. Fluctuation in the lean angle made by the tiltup model.

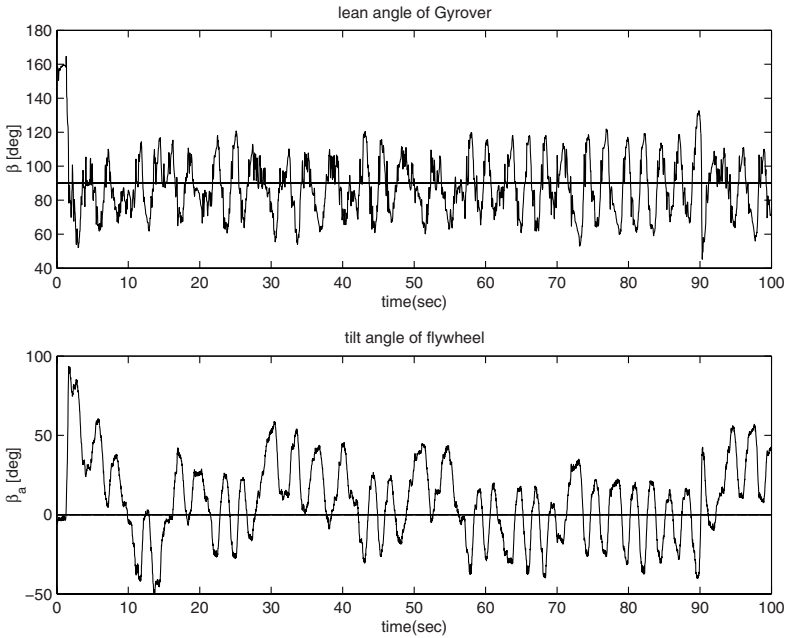


Fig. 5.25. Tiltup and vertical balanced motion by CNN models.

6

Shared Control

Based on the successful implementations of the lateral balancing and tiltup motion, in this chapter, we are going to develop a shared control framework for Gyrover. In fact, any situation of a system using shared control will involve human interactions. Under shared control, the human operator acts as a supervisor for overall control, while the robot itself can handle some local motions which in turn assist the human. In order to distribute the control tasks systematically, we develop an expression to make such a decision. Experimental results will be given in order to verify our idea.

6.1 Control Diagram

In fact, shared control happens in many daily examples, especially for human-animal interactions. First of all, let's consider the horse riding case [103], it is a fairly good example of a semi-autonomous system, or more specifically, a shared control system.

For the horse which is being ridden by a human, it is usually able to take care of all low-level tasks such as coordination of leg motion, stability, local obstacle avoidance and provide enough power and speed for different actions. On the other hand, the rider provides global planning, interacts with the horse to arrive at different locations and achieve various goals. At the same time, the rider can override any horse behavior by pulling the reins or hitting on the horse's body if necessary. Throughout the journey, the rider relies on the horse motoric abilities and the horse's behaviors become more intelligent by receiving the rider's commands. The interaction between the two individuals happens in a natural and simple way.

Another example we want to illustrate is to ask a robotic arm to handle a cup of tea [120]. The whole task can be decomposed into two subtasks: (i) to handle the cup of tea safely without pouring the tea (local balancing), and (ii) to reach the desired location (global navigation). In a teleoperated environment, it may be difficult for a human operator to perform both tasks

Y. Xu and Y. Ou: Control of Single Wheel Robots, STAR 20, pp. 151–173, 2005.

simultaneously, or it could be mentally taxing. However, if an autonomous module is introduced for the local stability of the cup, the operator in the control loop is only responsible for the navigation task, which greatly reduces the burden for the operators. Moreover, it is clear that the performance of the system would be much better and stable than being controlled by a single entity (human/machine).

Gyrover is a complex system not only in terms of the difficulties in deriving its mathematical model, but also in terms of its control by human operator. The robot can be controlled manually through a radio transmitter with two independent joysticks, one of them is assigned to control the drive motor, while the other one is assigned to control the tilt motor. Similar to a bicycle, Gyrover is a single track vehicle which is inherently unstable in its lateral direction. Therefore, different from controlling a quasi-static mobile robot, the human operator not only handles the global navigation for the robot, but also needs to pay attention to govern the lean angle of the robot simultaneously. Moreover, the highly coupling effect between the wheel and the internal flywheel also complicates the control of Gyrover. To this end, for such a complex system, instead of creating fully autonomous control, it is much more practical to develop a control method which can "share" the workload of the human operator.

Recently, shared control has been widely applied into many robotics man-machine systems, from health care [10, 103, 26, 2, 97, 20] to telerobotics [121, 64, 39, 120, 29]. For rehabilitation applications, a typical example is robotic wheelchairs. Although the wheelchair itself can provide a level of autonomy for the users, it is still desirable that the user can augment the control by the on-board joystick in some special situations (e.g. docking, passing through a doorway). A telerobotic system usually consists of a human operator and several autonomous controllers. A human operator usually interacts with the system in different ways. One of the important issues is to develop an efficient method to combine a human and machine intelligences so that the telerobotic system can perform tasks which cannot be done by either a human or autonomous controller alone [39]. In these shared control systems, the autonomous modules exist in the system to assist the human operator during navigation, in order to relieve the stress of the operators in a complex system. Usually, the human operator is responsible for some high-level control (e.g. global navigation), while the machine performs low-level control (e.g. local obstacle avoidance).

In fact, the two behaviors we have mentioned in the previous chapters, (i) Lateral balancing and (ii) Tiltup motion, are designed to tackle the robot's instability problem in the lateral direction. Since we have successfully modeled and implemented the two behaviors by a machine learning approach and verified them in experiments, the next step is to incorporate these motions with human control in order to develop a shared control framework for Gyrover. We prefer using a shared control scheme rather than a fully autonomous one because of the following reasons:

- **Sophisticated dynamic system.** As mentioned before, it is difficult for us to obtain a complete mathematical model to govern the motions of Gyrover, due to its complicated dynamic and nonholonomic nature. This makes us encounter many difficulties in developing a fully autonomous system for Gyrover at this stage.
- **Hardware limitations.** Due to the special physical structure of Gyrover, the current prototype of Gyrover we are using still does not have any navigation devices equipped on-board (e.g. vision), which makes it impossible for the robot to navigate itself.
- **Importance of human operators.** Practically, some complicated tasks, which may be trivial for humans, are often not performed well by robots. Therefore, a human operator is essential to exist in the control loop in order to monitor and operate the executive system.
- **Time and cost.** Building a fully autonomous system which provides safe and robust performance would be time consuming and costly, in terms of computations and resources. In contrast, it is far more practical and much cheaper to develop a semi-autonomous system.
- **Accuracy vs Reliablilty.** Machines are excellent in performing repetitive tasks quickly and accurately but their abilities to adapt to changes in the environment is low. On the other hand, humans are usually reliable, with tremendous perceptive ability and good decision making in unpredictable situations, but their accuracy is relatively lower than machines. Shared control can let them compensate each weakness which would result in better control.
- **Teleoperations.** Gyrover can be operated by humans through a radio transmitter, which allows humans to participate in the control of the robot.

The main difficulty in developing a shared control for Gyrover is due to the access to the tilt motor. Since the lean angle of the robot is controlled by the tilt motor, not only the autonomous module will access the tilt motor to achieve stability in the lateral direction, the human operator also needs to access the tilt motor during navigation. At a particular time instant, these commands may contradict each other. Therefore, it is a big issue to let the system decide which command is going to be executed, and at the same time, manage the contaminated commands in a reasonable way. To this end, we have developed an expression for making this decision, which will be discussed in the later part of this chapter. With better sharing between the machine and human operator, the performance of the system can be enhanced, and the range of tasks that can be performed by the system can also increase.

6.2 Schemes

In fact, there are many aspects of "sharing" in shared control, which vary from application to application. Basically, semi-autonomous control can be categorized into serial and parallel types [120]. In the serial type, manual control and

autonomous control cannot be executed simultaneously, only one of them will be selected at a time. In parallel type, both manual and autonomous control can be executed simultaneously.

In the following sections, we will briefly discuss three operating modes of shared control, namely: (1) Switch mode, (2) Distributed mode, and (3) Combined mode.

6.2.1 Switch Mode

In switch mode, manual and autonomous control are switched in serial, as shown in Figure 6.1. The condition to trigger the switch depends on applications, for example, if an operator acts as a supervisor of the control system, the human control will only be activated whenever the system reaches an "ill condition". No matter which control module is switched, the robot will be fully controlled by the selected one. If high cooperation between the machine and operator is required, we must have a function (Π) which can "smartly" switch between the two control modules.

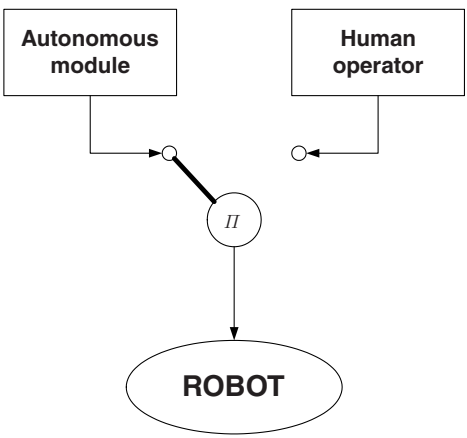


Fig. 6.1. Switch mode.

6.2.2 Distributed Mode

Figure 6.2 illustrates the architecture of distributed control. Different from switch mode, both manual and the autonomous control can be executed in parallel in this mode. The control of various actuators (u_i) in the entire system will be distributed to either of the two modules.

Therefore, the two entities can exist in the system peacefully without disturbing each other. However, this also shows the weakness of this mode

because there is no communication bewteen the two entities. The operator cannot modify the commands from autonomous module even if the robot performs or tends to perform undesirable motions.

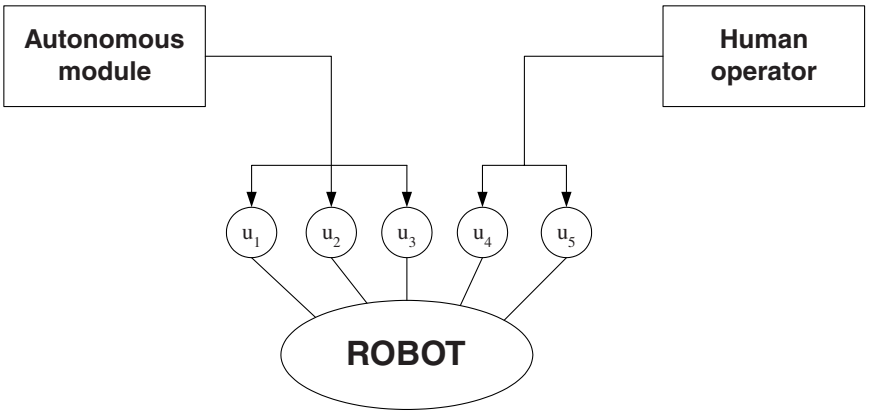


Fig. 6.2. Distributed control mode.

6.2.3 Combined Mode

Combined mode is in fact an extension of distributed mode, Figure 6.3. However, the input to a single actuator is a combination of the operator's command and the machine's command. There are many ways to combine the output vectors from the task modules: a simple summation, a simple average, weighted sum and average, voting on angle and velocity, and some unusual variations. In practice, the weighted average performs well since it is not computationally expensive and its performance is predictable [29].

6.3 Shared Control of Gyrover

Analogous to the example of handling a cup of tea, in our approach, in order to reduce the operator burden in controlling a statically unstable robot, it is desired that Gyrover itself can maintain a degree of local balancing, while the operator only responsible for the navigation task. In considering which mode of sharing is suitable for Gyrover shared control, we found that the commands from the automation module (lateral balancing and tiltup) always contradicts the navigation commands. It is due to the special steering mechanism of Gyrover, which is entirely contributed by the tilting effect of the internal flyhwheel.

As mentioned in section 3.2.1, when a disc is rolling, it will steer in the direction in witch it is leaning. Since the autonomous module is designed to keep

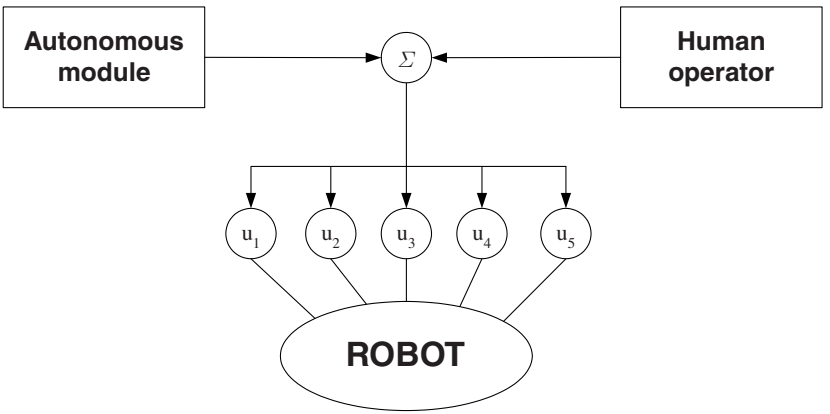


Fig. 6.3. Combined mode.

the lean angle in the vertical position, if we attempt to steer to the left/right manually (i.e. lean to the left/right), the machine will generate commands to stabilize the robot back to the vertical position, which will totally oppose the changes we want to make. Therefore, the commands from the two modules are impossible to combine into a single valid command during navigation.

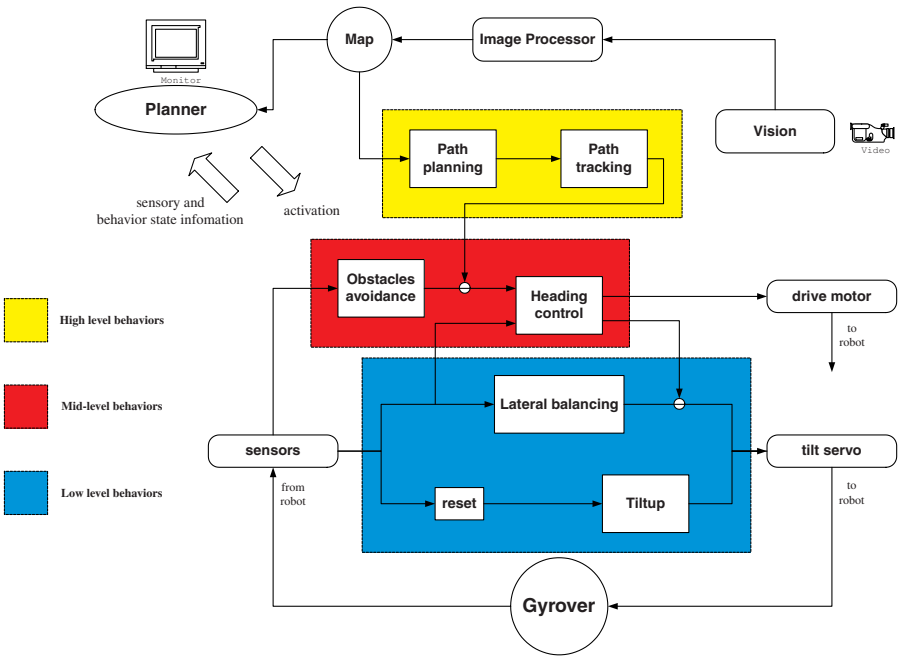


Fig. 6.4. A detailed structure of behavior connectivity in Gyrover control.

Referring to Figure 6.4, the mid and high-level behaviors are replaced by a human operator in shared control. Due to the high flexibility of the subsumption architecure, we obtain the shared control architecture as shown in Figure 6.5, without destorying the original control structure, which shows the beauty of behavior-based control architecture. Since the navigation tasks are entirely given to the human operator, the operator will solely control the drive motor through a radio transmitter. On the other hand, because the robot can maintain lateral stability when it stops rolling, or when a complete fall is detected, it will automatically tiltup back to its upright position. Thus, the tilt motor is jointly controlled by the operator and the machine.

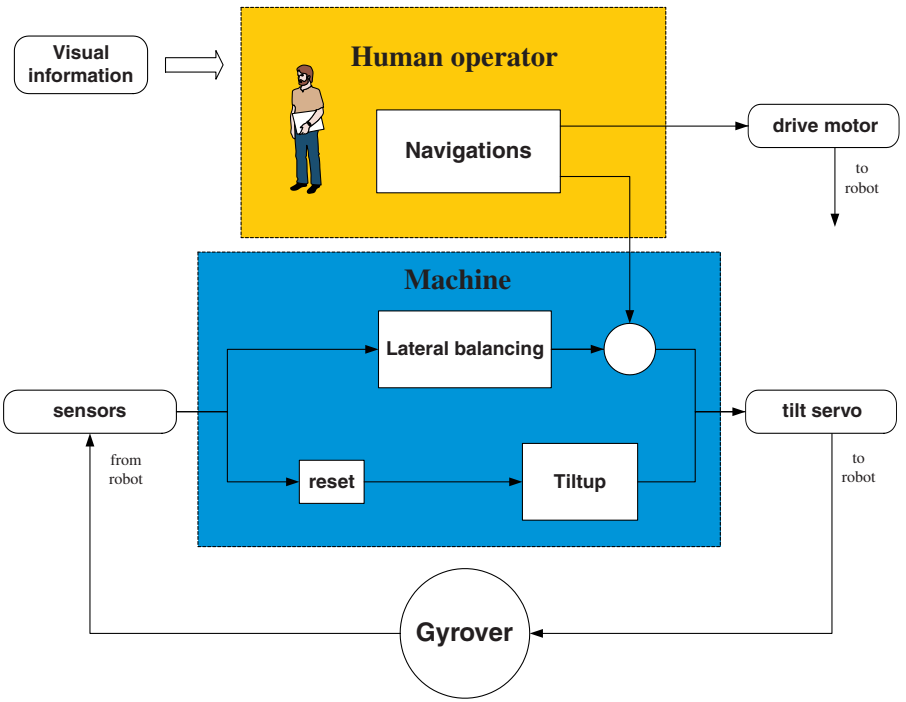


Fig. 6.5. Subsumption architecture of shared control.

According to Figure 6.5, regarding the tilt motor, switch mode is used since the operator and the machine cannot control the motor at the same time; regarding the whole structure, the system is somewhat in a distributed mode of sharing. As a result, the shared control of Gyrover combines the switch mode and the distributed mode, which compensates each mode's weakness.

6.4 How to Share

Recalling the horse riding example, it is believed that the horse acknowledges the rider's commands if they exceed a certain threshold. This threshold may depend on the horse's training (reliability of the autonomous system), the skill of rider, and on the situation at hand. If the rider wishes to correct or modify the horse current behavior, he/she will increase the level of stimulus which is acted on the horse (pulling the reins more or pushing harder on the saddle). This continues until the horse changes its behavior as wished by the rider. A poor communication or compromise between them can lead to undesirable or even dangerous results. Therefore, in this section, we develop a function to decide whether to follow or neglect the commands from the online operator.

First of all, let's introduce the variables that constitute the function, which are similar to those proposed in [103]:

1. **Degree of Autonomy**, A where $0 \leq A \leq 1$.
   This is a parameter which can be adjusted by the online operator. If the operator (a novice) wishes to rely much more on the autonomous module, he/she should select a higher value of A at the beginning of an operation. Otherwise, if an experienced operator is confident with his/her control skills, a lower value of A can be selected. We will demonstrate the effect of this parameter later.
2. **Strength of conflict**, S where $0 \leq S \leq 1$.
   This parameter measures the conflict between the operator and the current status of the system, it will vary from time to time whenever the operator is given a command to alter the system's trajectory. A high value of S indicates that the operator is making a control command which greatly affects the current status of the system, while a low value of S indicates that only a small disturbance is generated. This value will pass to the function instantaneous to make a decision whether to execute the operator's command or not. The strength of conflict S can be defined as:

$$S_\beta = \frac{\partial \beta}{\partial \beta_{\max}} \qquad \text{or} \qquad S_{\text{out}} = \frac{|u_{\text{operator}} - u_{\text{machine}}|}{\partial u_{\max}} \tag{6.1}$$

   where S_β is measured in terms of the changes in the lean angle β of the robot, S_{out} is in terms of the conflict between the command from the operator and the machine.
3. **Confidence level**, C where $0 \leq C \leq 1$.
   Contradictory to the strength of conflict, C is a parameter to show the confidence of an operator in making the current control command. It is obvious that the higher value of C, the more confident the operator is. This is also a time varying parameter which will pass to the function to let the system make a decision. The confidence level C can be defined as:

$$C = \frac{|\partial u_{\text{operator}}|}{\partial u_{\max}} \tag{6.2}$$

Based on the above definition, at a particular time instant, the system receives a command from the operator and the machine simultaneously, and we obtain the following relationship between S and C:

$$\begin{rcl} \text{if } C > S, & \text{follow the operator's command,} \\ C \leq S, & \text{follow the machine's command.} \end{rcl} \tag{6.3}$$

The above expressions imply that if the operator is confident enough to modify the current system trajectory, his/her command will be executed. On the other hand, if the system determines that the command of the operator is potentially to let the robot fall down, his/her command will be neglected, and the system will execute the balancing command from the autonomous module. However, the threshold of the above expressions remains constant and it is dependent on the system parameters. Practically, a system may be potentially operated by different operators, so it is desirable that the threshold of the decision be dependent on the operator. To this end, we introduce the parameter of Degree of Autonomy (A) into the above expressions,

$$\begin{rcl} \text{if } C \cdot (1 - A) > S \cdot A, & \text{follow the operator's command,} \\ C \cdot (1 - A) \leq S \cdot A, & \text{follow the machine's command.} \end{rcl} \tag{6.4}$$

By rewriting equation (6.4), we have,

$$\Pi(A, S, C) = \lambda \cdot C - S \tag{6.5}$$

where $\lambda = (1 - A)/A$ for simplicity, and the decision finally becomes,

$$\begin{rcl} \text{if } \Pi(A, S, C) > 0, & \text{follow the operator's command,} \\ \Pi(A, S, C) \leq 0, & \text{follow the machine's command.} \end{rcl} \tag{6.6}$$

The function Π is called a decision function which allows a system to decide whether to execute the command from an operator in a shared control environment. To validate the decision function, we let $A = 0$, which implies that the operator do not need any assistance from the autonomous module and the system should respond to all the commands from the operator. From equation (5.6),

$$\Pi(0, S, C) = +\infty > 0 \qquad \forall S, C$$

$\Pi(0, S, C)$ is always positive so that the system always executes the commands from the operator. Now, consider when $A = 1$,

$$\Pi(1, S, C) = -S \leq 0 \qquad \forall S, C$$

$\Pi(1, S, C)$ is always negative or equal to zero, which implies the system will totally follow the machine commands and disregard all the operator's command.

To further validate the decision function Π in (6.5), we perform the following experiments to see how the system works with this function. Tables 6.1, 6.2 and 6.3 show the values obtained from the decision function Π by using $A = 0.25$, $A = 0.50$ and $A = 0.75$ respectively. Based on the decision criteria in (6.6), if $\Pi(A, S, C)$ is greater than zero, the system will execute the operator's command at that particular moment, otherwise, the machine's command will be executed. In each table, a shaded value represents the system has chosen the operator's command.

Table 6.1. Decision making of $A = 0.25$.

	Current lean angle of the robot β										
∂u	40^o	50^o	60^o	70^o	80^o	90^o	100^o	110^o	120^o	130^o	140^o
0	-0.56	-0.44	-0.33	-0.22	-0.11	0	-0.11	-0.22	-0.33	-0.44	-0.55
2	-0.36	-0.25	-0.14	-0.03	0.08	0.11	-0.01	-0.12	-0.23	-0.34	-0.45
4	-0.17	-0.06	0.06	0.17	0.28	0.21	0.10	-0.01	-0.12	-0.23	-0.34
6	0.03	0.14	0.25	0.36	0.43	0.32	0.21	0.09	-0.02	-0.13	-0.24
8	0.22	0.33	0.44	0.56	0.53	0.42	0.31	0.20	0.09	-0.02	-0.13
10	0.42	0.53	0.64	0.75	0.64	0.53	0.42	0.31	0.19	0.08	-0.03
15	0.09	1.01	1.13	1.01	0.90	0.79	0.68	0.57	0.46	0.35	0.24
20	1.39	1.50	1.39	1.28	1.17	1.06	0.94	0.83	0.72	0.61	0.50
30	2.14	2.03	1.92	1.81	1.69	1.58	1.47	1.36	1.25	1.14	1.03
40	2.67	2.56	2.44	2.33	2.22	2.11	2.00	1.89	1.78	1.67	1.56

Table 6.2. Decision making of $A = 0.50$.

	Current lean angle of the robot β										
∂u	40^o	50^o	60^o	70^o	80^o	90^o	100^o	110^o	120^o	130^o	140^o
0	-0.56	-0.44	-0.33	-0.22	-0.11	0	-0.11	-0.22	-0.33	-0.44	-0.55
2	-0.46	-0.35	-0.24	-0.13	-0.02	0.01	-0.11	-0.22	-0.33	-0.44	-0.55
4	-0.37	-0.26	-0.14	-0.03	0.08	0.01	-0.01	-0.21	-0.32	-0.43	-0.54
6	-0.27	-0.16	-0.05	0.06	0.13	0.02	-0.09	-0.21	-0.32	-0.43	-0.54
8	-0.18	-0.07	0.04	0.16	0.13	0.02	-0.09	-0.20	-0.31	-0.42	-0.53
10	-0.08	0.03	0.14	0.25	0.14	0.03	-0.08	-0.19	-0.31	-0.42	-0.53
15	0.15	0.26	0.38	0.26	0.15	0.04	-0.07	-0.18	-0.29	-0.40	-0.51
20	0.39	0.50	0.39	0.28	0.17	0.06	-0.06	-0.17	-0.28	-0.39	-0.50
30	0.64	0.53	0.42	0.31	0.19	0.08	-0.03	-0.14	-0.25	-0.36	-0.47
40	0.67	0.56	0.44	0.33	0.22	0.11	0	-0.11	-0.22	-0.33	-0.44

Table 6.3. Decision making of $A = 0.75$.

	Current lean angle of the robot β										
∂u	40^o	50^o	60^o	70^o	80^o	90^o	100^o	110^o	120^o	130^o	140^o
0	-0.55	-0.44	-0.33	-0.22	-0.11	0	-0.11	-0.22	-0.33	-0.44	-0.55
2	-0.49	-0.38	-0.27	-0.16	-0.05	-0.03	-0.14	-0.25	-0.36	-0.47	-0.58
4	-0.43	-0.32	-0.21	-0.10	0.01	-0.06	-0.17	-0.28	-0.39	-0.50	-0.61
6	-0.37	-0.26	-0.15	-0.04	0.03	-0.08	-0.19	-0.31	-0.42	-0.53	-0.64
8	-0.31	-0.20	-0.09	0.02	0	-0.11	-0.22	-0.33	-0.44	-0.56	-0.67
10	-0.25	-0.14	-0.03	0.08	-0.03	-0.14	-0.25	-0.36	-0.47	-0.58	-0.69
15	-0.10	0.01	0.13	0.01	-0.10	-0.21	-0.32	-0.43	-0.54	-0.65	-0.76
20	0.06	0.17	0.06	-0.06	-0.17	-0.28	-0.39	-0.50	-0.61	-0.72	-0.83
30	0.14	0.03	-0.08	-0.19	-0.31	-0.42	-0.53	-0.64	-0.75	-0.86	-0.97
40	0	-0.11	-0.22	-0.33	-0.44	-0.56	-0.67	-0.78	-0.89	-1.00	-1.11

When $A = 0.25$, the system will be more likely to rely on the operator's command. In Table 6.1, most of the operator's commands are chosen even when the Confidence level of his/her control is quite low (smaller ∂u). On the other hand, for a higher value of A (Table 6.3), the system relies on the machine's commands more so that the frequency of accepting the operator's commands reduces significantly. The above experiments simply illustrate that the decision function Π can judge whether to execute a human operator's commands effectively by taking the value A into accounts, which is very important in a shared control system.

In fact, the system neglects the operator's commands only when the command is potentially dangerous to the robot. Since a positive change in the tilt command will give a positive change in the lean angle of the robot, if the lean angle is beyond 90^o, a larger ∂u will make the lean angle grow bigger, which potentially makes the robot falls down. Therefore, in this case, if the operator is not confident enough to make this change, his/her command will be neglected.

6.5 Experimental Study

In this section, we implement the shared control framework as shown in Figure 6.5, by applying the decision function we have mentioned in the last section. We have designed several tasks for the robot to perform under the shared control scheme, including (i) heading control (ii) straight path tracking, (iii) circular path tracking, and (iv) point-to-point navigation.

Since the autonomous module now in hand is only responsible for the lateral stabilization and tiltup motion when the robot is held in a stationary location, the navigation task of the robot will be entirely given to the human operator to control, which implies that the human cannot rely on the machine

throughout the navigation. Based on this limitation, we use a relatively high level of autonomy ($A \approx 0.25$) in Gyrover shared control. From the experiments, we can observe that even the operator has shared a level of control with the system, and the robot can still achieve some basic goals in mobile teleoperations.

6.5.1 Heading Control

The purpose of this experiment is to illustrate the cooperation between the human operator and the autonomous module in a shared control environment. One special feature of Gyrover is the ability to turn into a desirable heading direction at a stationary location, this motion can be achieved by controlling the lean angle of the robot (left/right) until the desired heading direction is reached.

When the robot is not rolling, the system will automatically execute the lateral balancing module in order to maintain its lateral stability, by controlling the tilt motor. If the operator wishes to command the robot to turn into a particular heading angle, he/she needs to make the robot lean at a certain angle by controlling the tilt motor; also, in this case, the robot must stop the autonomous module and execute the operator's command. Therefore, if the system cannot make the right decision, the operator can never control the robot to turn into a desired heading direction.

The result of using $A = 0.2$ and $A = 0.8$ in the heading control test is shown in Figure 6.6 and Figure 6.7 respectively. For $A = 0.2$, the operator triggers the control of the tilt motor at $7.5 \leq t \leq 9.5$ and $14.5 \leq t \leq 17$, in order to make the robot lean to a particular heading angle. It is clear that the operator augments the control in these periods successfully, which is expected when a low degree of autonomy is used. When there is no command from the operator, the robot will execute the lateral balancing control from the autonomous module in order to keep the robot around 90^o. For $A = 0.8$, the control trajectory of the operator is completely different from the final control output to the system. The operator wants to trigger the tilt motor, but the system neglects most of his/her commands and continues to execute the lateral balancing commands from the autonomous module. The system will only execute those commands from the operator only when the particular command greatly contributes to keeping the robot at 90^o, or when the confidence level is high, for instance, at $t \approx 13$ and $t \approx 17$.

6.5.2 Straight Path

In the straight path test, the operator is asked to control the robot to travel a straight path, approximately 44 ft long. The experimental setup is shown in Figure 6.8. Three trails are given in this experiment, the trajectory that the robot has travelled in each trail is shown in Figure 6.11. The sensor data of the robot in trail #3 is plotted in Figure 6.12.

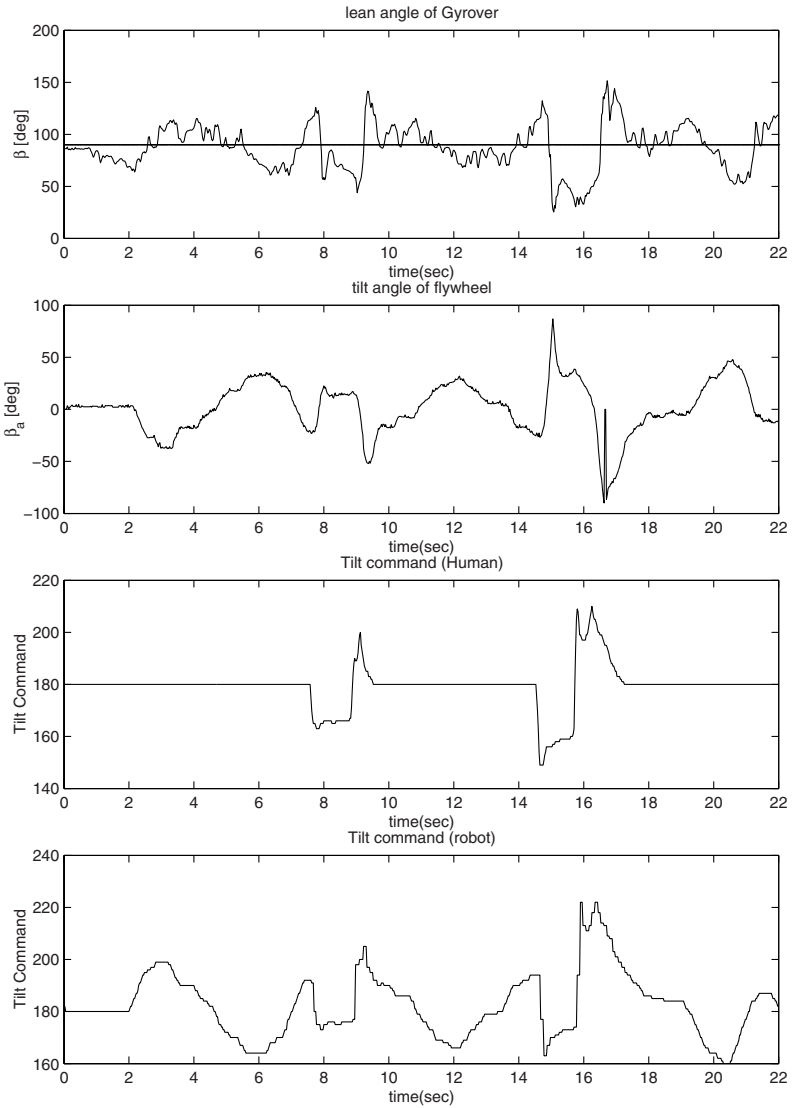


Fig. 6.6. Sensor data acquired in the heading control test, $A = 0.2$.

Under a shared control, although some of the control commands are ignored by the system (flattened peaks in the final output of tilt motor command), the operator is still able to control the robot to travel a nearly straight path, with an average 0.1736 ft offset from the desired path. At $t = 9$, the robot received no commands from the operator and started to execute the lateral balancing module to balance the robot. As mentioned earlier, the control of the drive motor is entirely given to the operator, therefore, the system

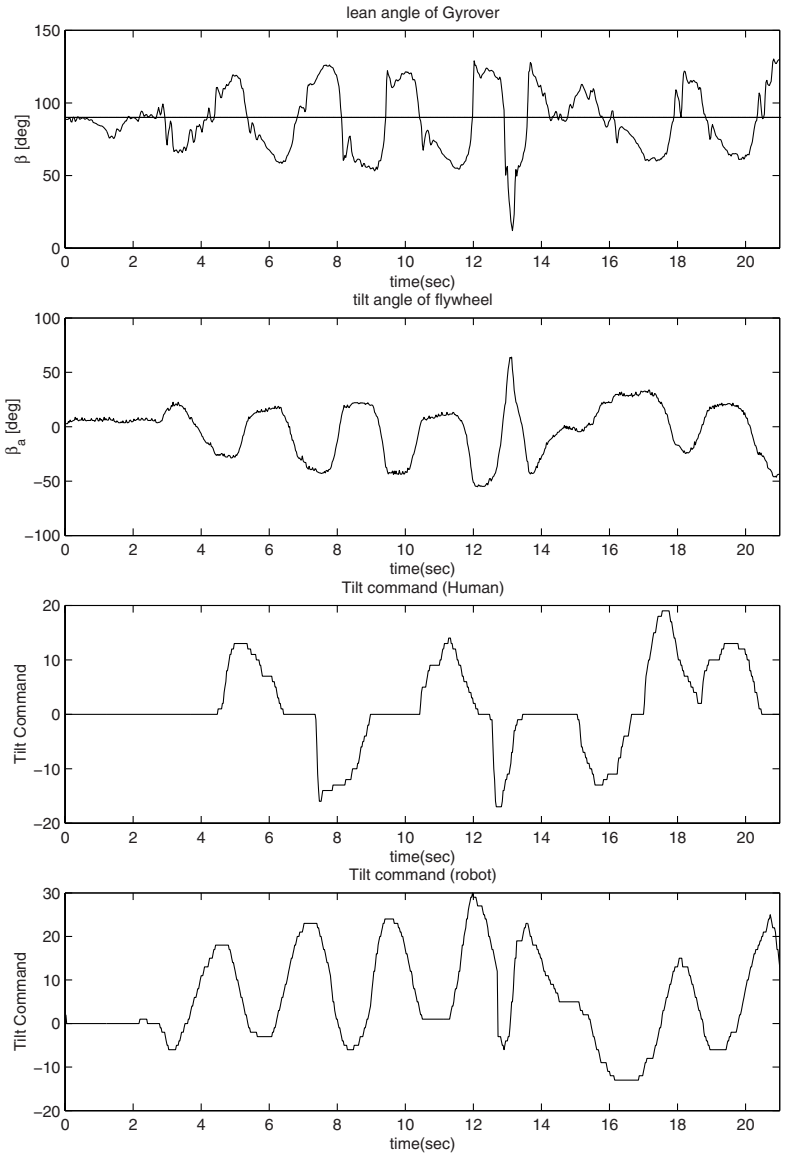


Fig. 6.7. Sensor data acquired in the heading control test, $A = 0.8$.

will not interfere with the drive motor command, which directly follows the control of the operator.

Fig. 6.8. Experiment on tracking a straight path under shared control.

6.5.3 Circular Path

Similar to the straight path test, the experimental setup is shown in Figure 6.9. This time, the operator is required to control the robot travel a circular path. In order to make the robot to turn in place, the operator needs to the tilt the internal flywheel to make a "lean steering" precisely. If the robot fails to follow the right commands, it is unable to steer well. Figure 6.13 indicates the desired path and the actual path travelled by the robot respectively. Figure 6.14 shows the corresponding sensor data (trail #3) of the robot during travelling in a circular path.

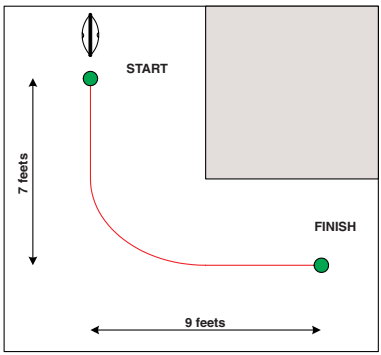


Fig. 6.9. Experiment on tracking a curved path under shared control.

The average offset in the circular path test is 0.51 ft. Although the robot cannot track the circular path precisely, the operator can control the robot to move back to the target location within 0.25 ft at nearly the end of the experiments. Therefore, with a degree of shared control with the robot, the operator is still able to control the robot to turn a tight corner.

6.5.4 Point-to-point Navigation

In this experiment, we require the robot to travel from one location to another which are separated by a right-hand corner and they are far apart ($\approx$ 60 ft),

Figure 6.10. The operator needs to control the robot to move from a starting area to a specific destination, which is a 2 ft $\times$ 2 ft region (the dimension of Gyrover is about 1.5 ft $\times$ 0.8 ft as viewed from the top). This experiment has two main goals:

1. The robot must reach the destination within the specific area.
2. After the robot has reached the destination, it is required that the robot can maintain its lateral balance even when the operator does not further control it.

The experimental results are shown in Figure 6.15 and 6.16.

Although we are not concerned with whether the robot can accurately track the path or not, the overall offset from the path is 1.18 ft, which is an acceptable value for a 60 ft long journey. Moreover, for the three trails in this experiment, all the trajectories of the robot are converging to the destination at the end of the path. From Figure 6.16, when $t \geq 14$ (at the destination), the operator did not command the robot anymore, however, the robot can balance itself at around 90^o. Therefore, under a shared control environment, with the human operator responsible for the navigation task of the robot, the robot is able to move from one location to another location which are far apart, and to balance itself at the vertical position when the robot stops moving (with no operator's command).

6.6 Discussions

From the results we obtained from the previous experiments, we verify that our proposed shared control algorithm can let the system choose between human operator's control commands or the commands from the autonomous module systematically. Whenever the operator has chosen a high level of autonomy, the system will execute the command from the the autonomous module unless the operator has given a command which is 'confident' enough to overcome the conflict between the operator and the machine. On the other hand, if a low degree of autnomy is chosen, the system will follow the operator's command unless a 'significant' error/conflict is measured. The proposed shared control algorithm is able to allow two entities (human and machine) to exit in the same system simultaneously.

Although Gyrover does not have an autonomous module to navigate itself to travel from one location to another, this can be done by sharing the navigation task with the operator. Under shared control, the robot will maintain its lateral balance when the operator does not command it. On the other hand, under a degree of sharing, the operator is still able to control the robot to do some specific tasks (straight path tracking, point-to-point navigation, etc). It is believed that if an autonomous navigation module exists in the system, the operator can share more naviagtion control to the machine using the

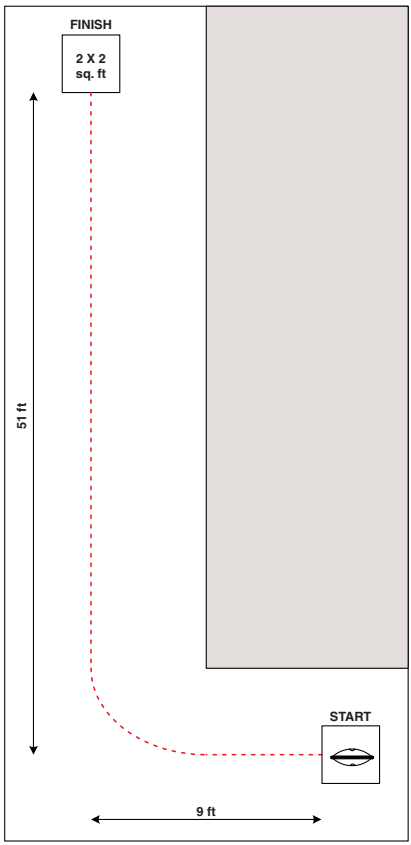


Fig. 6.10. Experiment on point-to-point navigation under shared control.

proposed shared control algorithm, which can greatly reduce the duty of the online human operator.

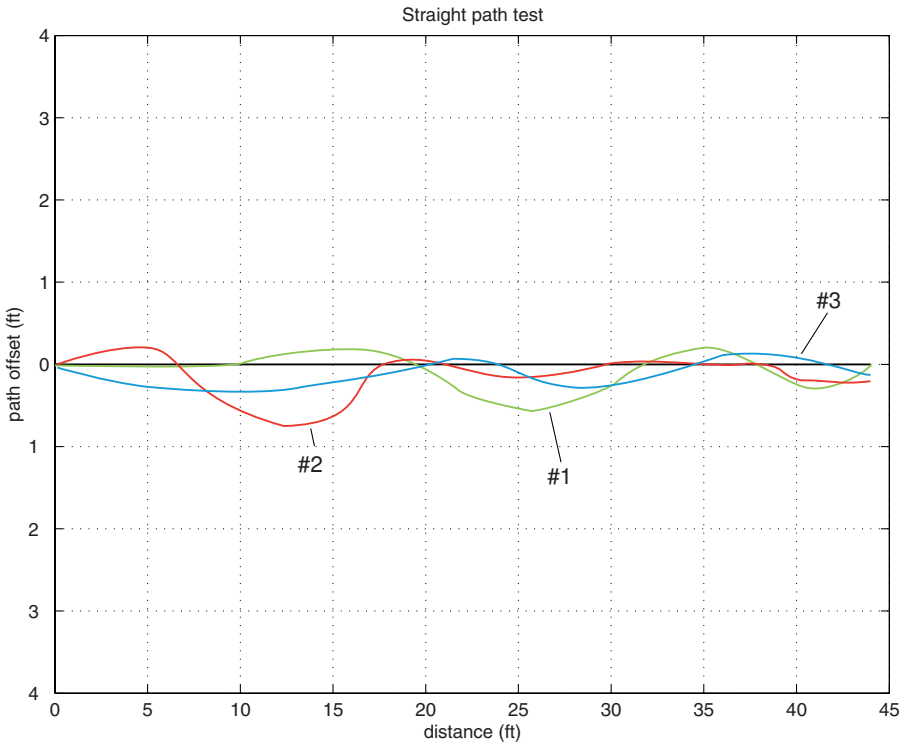


Fig. 6.11. Trajectory travelled in the straight path test.

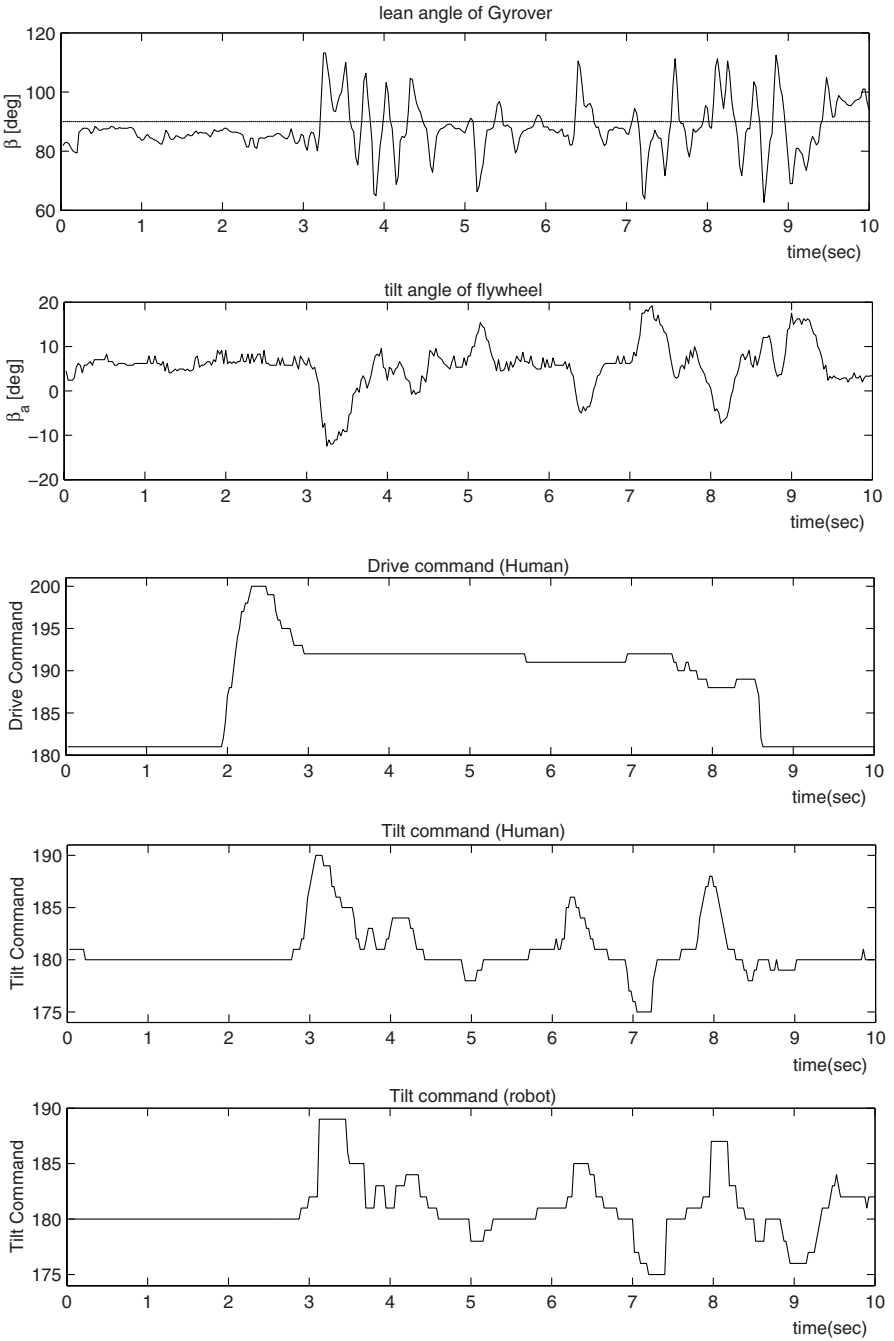


Fig. 6.12. Sensor data acquired in the straight path test.

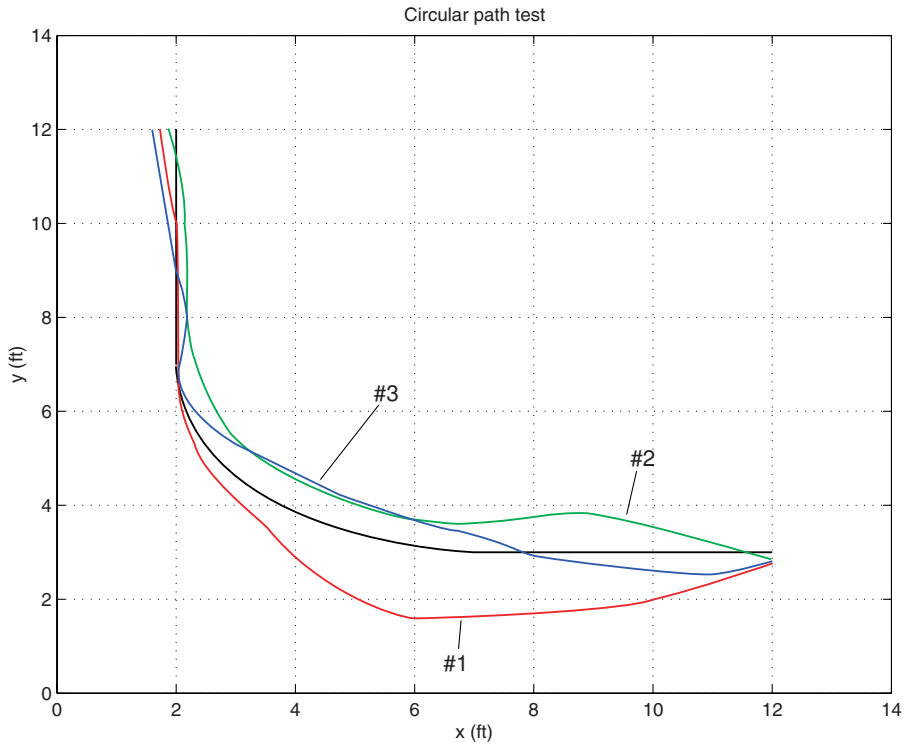


Fig. 6.13. Gyrover trajectories in the curved path test.

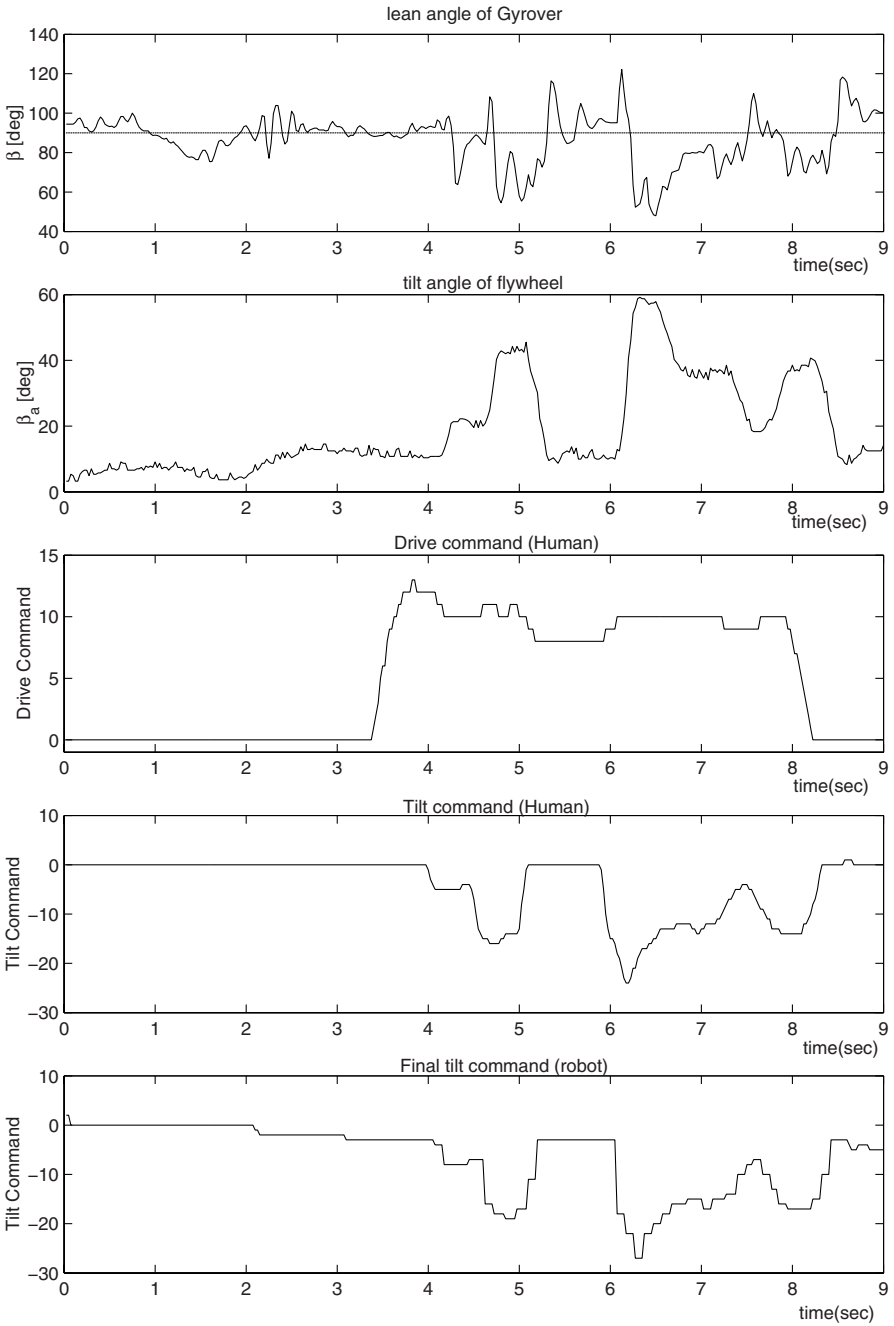


Fig. 6.14. Sensor data acquired in the circular path test.

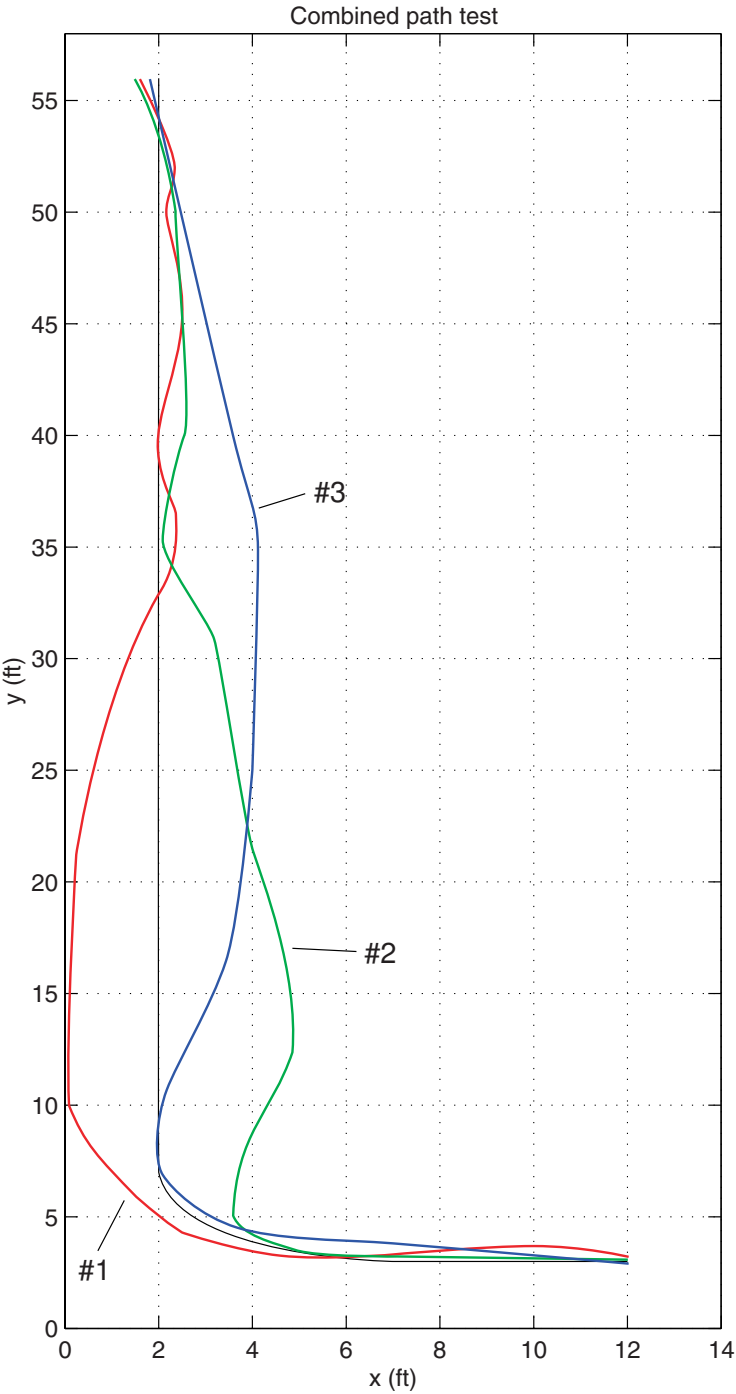


Fig. 6.15. Gyrover trajectories in the combined path test.

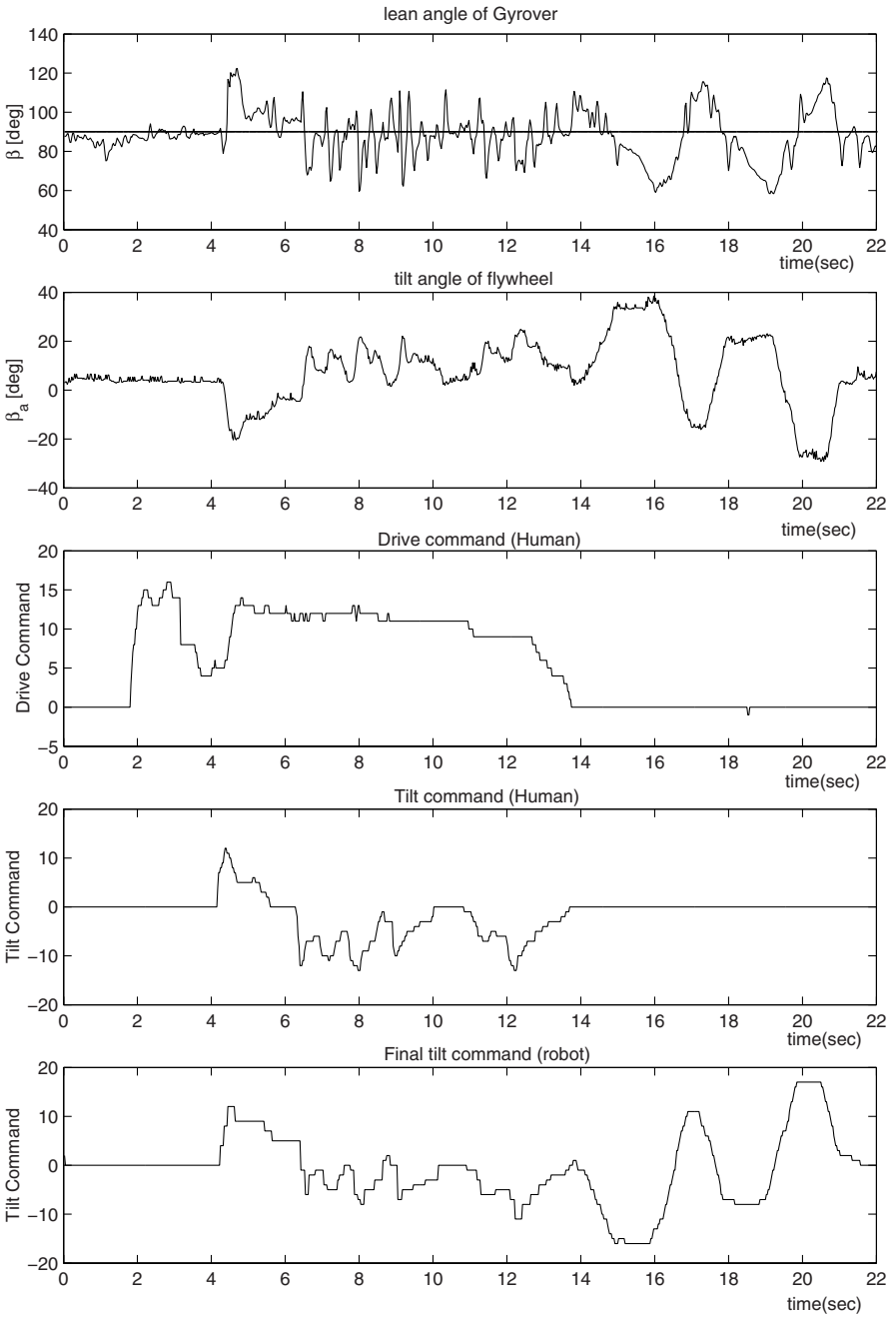


Fig. 6.16. Sensor data acquired in the combined path test.

7

Conclusions

7.1 Concluding Remarks

7.1.1 Concept and Implementations

Gyrover is a novel concept for mobility that has distinct advantages over conventional, statically stable vehicles. The concept has been verified with three working models. Gyrover is particularly suited for operations at high speed and rough terrain, and also shows promise as a marine/amphibious vehicle. The mechanism can be completely enclosed, presenting a very clean, protected envelope.

During the past several years, we have made significant advances in the technologies for a gyroscopically stabilized wheel. We have gained an understanding of the basics of gyro-stabilization, and have demonstrated the feasibility of the Gyrover concept.

7.1.2 Kinematics and Dynamics

We have developed the kinematic and dynamic models of Gyrover in a horizontal plan. In derivation of the model, we consider the robot as a combination of three components: a rolling disk, an internal mechanism and a flywheel. They are linked by a two-link manipulator. We simplified the model by decoupling the tilt angle of the flywheel from the dynamics. We verified the model by experiment and simulation. We demonstrated that the dynamics of the robot is nonholonomic and underactuated. We demonstrated the dynamics coupling between the wheel and the flywheel, through the stabilization and tilting effect of the flywheel on the robot.

Based on the above work, we have established the dynamics of a robot rolling without slipping on an inclined plane. The pendulum swinging motion is neglected and the vertical offset of the actuation mechanism from the axis of the whole wheel is reduced from the dynamics. If we let l_1, l_2 and θ be zero, the dynamics are exactly the same as Gyrover on a horizontal plan.

Y. Xu and Y. Ou: Control of Single Wheel Robots, STAR 20, pp. 175–177, 2005.

7.1.3 Model-based Control

First, using the linearization method, we developed a linear state feedback approach to stabilize the robot at any desired lean angle. This feedback provides means for controlling the steering velocity of the robot. Finally, we developed a line following controller for tracking any desired straight line while keeping balance. The controller is composed of two parts: the velocity control law and the torque control law. In the velocity control law, the velocity input (steering velocity) is designed for ensuring the continuity of the path curvature. Then, the robot can be stabilized for tracking a lean angle trajectory in which the steering velocity is identical to the desired value. We addressed the effects of the initial heading angle, the rolling speed and the controller gains on the performance of the controller. The work is of significance in developing automatic control for the dynamically stable but statically unstable robot.

Then, we studied control problems for a single wheel, gyroscopically stabilized robot. We investigated the dynamics of the robot system, and analyzed the two classes of nonholonomic constraints of the robot. We proposed three control laws for balance, point-to-point control and line tracking. The three problems considered are fundamental tasks for Gyrover control.

7.1.4 Learning-based Control

First, we proposed a novel method for human control strategy learning for a dynamically stable system using an SVM. The SVM is a powerful tool for function estimation and is feasible to characterize the stochastic process in human control strategy learning. Moreover, we proposed approaches to formulate conditions for the SVM learning control closed-form system to be stable and to determine the domain of attraction, if it is stable.

Second, we proposed a new method of adding new training samples without increasing costs. We use the local polynomial fitting approach to individually rebuild the time-variant functions of system states. Through interpolating, we can effectively produce any number of additional training samples. A number of experiments based on a dynamically stable system – Gyrover – show that by using additional unlabelled samples, the learning control performance can be improved, and therefore the *overfitting* phenomenon can be mitigated.

Third, we examine a method of selecting input variables, through the evaluation of the significance order function and dependent analysis to enhance the learning control performance. It was observed that by executing the previous input selection process in the modeling process, the '*curse of dimensionality*' phenomenon might be mitigated. The mitigation of the '*curse of dimensionality*' phenomenon is important in a learning control process, especially that of a dynamically stable system like Gyrover, for the following reasons. First, the initial condition may differ from case to case. Second, the controllers and the system states are combined to form an integrated system and interact with

each other. Third, if the learning model does not effectively set up the mapping between the system states and the control inputs, the learning controllers from the model will be rigid and fail to make the control process converge to the targets.

Last, we develop a shared control framework for Gyrover, based on the behavior-based architecture. Since building a fully autonomous system is costly and sometimes not practical, the main purpose of shared control is to reduce the operator's control burden in a complex system. In order to distribute the workload systematically in a shared control environment, we develop a decision function to let the system judges whether to execute the operator's command or not, by considering the Degree of Autonomy, the Strength of Conflict and the Confidence level. Experiments show that this shared control framework is able to share some of the control tasks from the operator without decreasing the maneuverability of the robot.

7.2 Future Research Directions

While this book provides a great deal on mechanism design and control implementation of Gyrover, there are a number of different directions in which the area in this work can be extended and applied. The followings are some possible improvements and extension of this work.

We proposed the foundations of the shared control of Gyrover. It is only the first step towards full implementation. We hope that the learning controllers are not only to be shared with a human operator, but also to be shared with the model-based control.

We are seeking the possibility of applying these technologies on other dynamically stable systems, such as inverted pendulum systems, rolling disks, bicycles, biped robots, hopping robots and so on. We believe that these technologies are suitable in overcoming the engineering challenges for the control and design of similar systems as Gyrover.

References

1. M. Aicardi, G. Casalino, A. Balestrino, and A. Bicchi, "Closed loop smooth steering of unicycle-like vehicles," *Proc. of the 33th IEEE Conf. on Decision and Control*, vol. 3, pp. 2455 -2458, 1994.
2. P. Aigner and B. McCarragher, "Shared control framework applied to a robotic aid for the blind," *Proc. of the 1998 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 717-722, 1998.
3. H. Arai and S. Tachi, "Position control of a manipulator with passive joints using dynamic coupling", *IEEE Trans on Robotics and Automation,* vol.7, no.4, pp.528-534, 1991.
4. H. Asada and S. Liu, "Transfer of human skills to neural net robot controllers," *Proc. of the 1999 IEEE International Conference on Robotics and Automation*, 1991.
5. K. W. Au, "Dynamics and control of a single wheel, gyroscopically stabilized robot," M.Phil. Thesis, The Chinese University of Hong Kong, 1999.
6. K. W. Au and Y. S. Xu, "Decoupled dynamics and stabiliztion of single wheel robot," *Proc. of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems* , vol. 1, pp. 197-203, 1999.
7. K. W. Au and Y. S. Xu, "Path following of a single wheel robot," *Proc. of the 2000 IEEE International Conference on Robotics and Automation,* vol. 3, pp. 2925-2930, 2000.
8. A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Trans. on Information Theory,* vol. 39, no. 3, pp. 930-945, 1993.
9. R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Trans. on Neural Networks,* vol. 5, no. 4, pp. 537-550, 1994.
10. D. A. Bell, S. P. Levine, Y. Koren, L. A. Jaros, and J. Borenstein, "Design criteria for obstacle avoidance in a shared-control system," *Whitaker Student Scientific Paper Competition, RESNA'94 Annual Conference*, June, 1994.
11. R. Bellman, *Adaptive control processes: a guided tour.* New Jersey: Princeton University Press, 1961.
12. M. G. Bekker, *Theory of land locomotion,* University of Michigan Press, pp.213-217, 1956.
13. M. G. Bekker, "Accomplishments and future tasks in off-road transportation", *Journal of Terramechanics,* vol.11, no.2, pp.11-30, Permagon Press, Ltd., 1974.

14. M. Bergerman and Y. Xu, "Dynamic coupling Index for underactuated manipulators," *Journal of Robotic Systems,* vol.12, no.10, pp.693-707, 1995.
15. S. Bernhard, C. J. C. Burges, and A. Smola, *Advanced in kernel methods support vector learning*, Cambridge, MA, MIT Press, 1998.
16. A. V. Beznos, et. al., "Control of autonomous motion of two-wheel bicycle with gyroscopic stabilisation," *Proc. IEEE Int. Conf. on Robotics and Automation,* vol. 3, pp. 2670-75, 1998.
17. C. Bishop, "Exact calulation of hessian matrix for the multilayer perceptron," *Neural Computation,* vol. 4, pp. 494-501, 1992.
18. A. M. Bloch, M. Reyhanoglu, and N. H. McClamroch, "Control and stabilization of nonholonomic dynamic systems," *IEEE Transaction on Automation Control*, vol. 37, no. 11, pp. 1746-1756, 1992.
19. A. L. Blum and P. Largely, "Selection of relevant features and examples in machine learning," *Artif. Intell.,* vol. 97, pp. 245-271, 1997.
20. G. Bourhis and Y. Agostini, "Man-machine cooperation for the control of an intelligent powered wheelchair," *Journal of Intelligent and Robotics Systems,* vol. 22, no.3-4 pp. 269-287, 1998.
21. H. B. Brown and Y. Xu, "A single wheel gyroscopically stabilized robot," *Proc. IEEE Int. Conf. on Robotics and Automation,* vol. 4, pp. 3658-63, 1996.
22. M. Buhler, D. E. Koditschek, and P. J. Kindlmann, "A simple juggling robot: theory and experimentation," Experimental Robotics, edited by V. Hayward and O. Khatib, pp.35-73, Springer-Verlag, 1989.
23. C. J. C. Burges and B. Scholkopf, "Improving the accuracy and speed of support vector learning machines," In M. Mozer, M. Jordan and T. Petsche Editors, *Advances in neural information processing Systems 9.* Cambridge, MA, MIT Press, pp. 375-381, 1997.
24. G. Chryssolouris, M. Lee, and A. Ramsey, "Confidence interval prediction for neural network models," *IEEE Trans. on Neural Networks,* vol. 7, no. 1, pp. 229-232, 1996.
25. T. Cibas, F. F. Soulie, P. Gallinari and S. Raudys, "Variable selection with neural networks," *Neural Computing,* vol. 12, pp. 223-248, 1996.
26. R. A. Cooper, "Intelligent control of power wheelchairs," *IEEE Engineering in Medicine and Biology Magazine*, vol.14, issue: 4, pp. 423-431, 1995.
27. N. Cristianini and J. Shawe-Taylor, *A Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press, Cambridge, 2000.
28. G. Cybenko, "Approximations by suppositions of s sigmoidal function," *Mathematics of Control, Signal and Systems,* no. 2, pp. 871-875, 1989.
29. A. Douglas and Y. Xu, "Real-time shared control system for space telerobotics," *Proc. of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems '93, IROS '93*, vol. 3, pp. 2117-22, 1993.
30. D. R. Freitag, "History of wheels for off-road transport," *Journal of Terramechanics,* vol. 16, no. 2, pp. 49-68, Permagon Press, Ltd., 1979.
31. S. S. Ge, C. C. Huang, and T. Zhang, "Adaptive neural network control of nonlinear systems by state and output feedback," *IEEE Trans. on Systems, Man, and Cybernetics–Part B: Cybernetics,* vol. 29, no. 6, pp. 818-828, 1999.
32. S. S. Ge, G. Y. Li, and T. H. Lee, "Adaptive NN control for a class of strict-feedback discrete-time nonlinear systems," *Automatica,* vol. 39, pp. 807-819, 2003.

33. R. Genesio, M. Tartaglia, and A. Vicino, "On the estimation of asymptotic stability regions: stae of the art and new proposals," *IEEE Transactions on Automatic Control,* vol. 30, no. 8, pp. 747-755, 1985.
34. N. H. Getz, "Dynamic inversion of nonlinear maps with applications to nonlinear control and robotics," Ph.D. thesis, University of California at Berkeley, 1995.
35. N. H. Getz, "Control of balance for a nonlinear nonholonomic non-minimum phase model of a bicycle," *Proc. ACC,* vol. 1, pp. 148-151.
36. C. G. Gingrich, D. R. Kuespert, and T. J. McAvoy, "Modeling human operators using neural networks," *ISA Trans.,* vol. 31, no. 3, pp. 81-90, 1992.
37. T. Glad and L. Ljung, Eds., *Comtrol Theory: Multivariable and Nonlinear Methods*, London and New York : Taylor & Francis, 2000.
38. F. A. Graybill, *Theory and Application of Linear Model*, North Scituate, MA: Duxbury Press, 1976.
39. C. Guo, T. Tarn, N. Xi, and A.K. Bejczy, "Fusion of human and machine intelligence for telerobotic systems," *Proc. of the 1995 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3310-15, 1995.
40. Y. Hamamoto, S. Uchimura, T. Kanaoka, and S. Tomita, "Evaluation of artificial neural network classifiers in small sample size situations," *Proc. Int. Conf. Neural Networks*, pp. 1731-1735, 1993.
41. Y. Hamamoto, Y. Mitani, and S. Tomita, "On the effect of the noise injection in small training sample size situations," *Proc. Int. Conf. Neural Information Processing*, vol. 1, pp. 626-628, 1994.
42. C. Hautier, "Gyroscopic device for stabilization of laterally unstable vehicles," U.S. Patent, #3,787,066, 1974.
43. R. C. Hemmings, "Improvement in velocipede," U.S. Patent, #92,528, 1869.
44. J. K. Hodgins and M. H. Raibert, "Adjusting step length for rough terrain locomotion," *IEEE Transactions on Robotics and Automation,* vol.7 no.3, pp.289-298, 1991.
45. I. C. Holm,"Articulated, wheeled off-road vehicles," *Journal of Terramechanics,* vol. 7, no. 1, pp. 19-54, Permagon Press, Ltd., 1970.
46. G. Indiveri, "Kinematic time-invariant control of a 2D nonholonomic vehicle," *Proc. IEEE Conf. Decis. Control*, vol. 3, pp. 2112-2117, 1999.
47. E. G. John, *Modern Mechanix and Inventions*, June, 1935.
48. Y. J. Kanayama and F. Fahroo, "A new line tracking method for nonholonomic vehicles," *Proc. IEEE Int. Conf. on Robotics and Automation,* vol. 4, pp. 2908-11, 1997.
49. Y. J. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for a nonholonomic mobile robot," *Proc. IEEE/RSJ Int. Workshop on Intelligent Robots and Systems,* vol. 3, pp. 1236-41, 1991.
50. J. Karhunen and J. Joutsensalo, "Generalizations of principal component analysis, optimization problems, and neural networks," *Neural Computing,* vol. 8, pp. 529-562, 1995.
51. J. Karhunen, L. Wang and R. Vigario, "Nonlinear PCA type approaches for source separation and independent component analysis," *Proc. 1995 International Conference on Artificial Neural Networks*, vol. 2, pp. 995-1000, 1995.
52. A. Kemurdjian, et al, "Small Marsokhod configuration," *Proc. IEEE Int. Conf. on Robotics and Automation,* vol. 1, pp. 165-68, 1992.

53. Y. H. Kim and F. L. Lewis, "Neural network output feedback control of robot manipulators," *IEEE Transactions on Robotics and Automation,* vol. 15, no. 2, pp. 301-309, 1999.
54. K. Kira and L. A. Rendell, "The feature selection problem: traditional methods and a new algorithm," *Proc.* 10^{th} *National Conference on Artificial Intelligence*, pp. 129-134, 1992.
55. P. R. Klarer, "Recent developments in the robotics all terrain lunar explorer rover (RATLER) program," *ASCE Specialty Conference on Robotics for Challenging Environments,* Albuquerque, NM, 1994.
56. R. Kohavi and H. J. George, "Wrappers for feature subset selection," *Artif. Intell.,* vol. 97, pp. 273-324, 1997.
57. I. Kolmanovsky and N. H. McClamroch, "Development in nonholonomic control problems," *IEEE Contr. Syst. Mag.*, vol. 15, pp. 20-36, Jan., 1995.
58. I. Kolmanovsky and N. H. McClamroch, "Application of integrator backstepping to nonholonomic control problems," *Proc IFAC Nonlinear Control Syst. Des. Symp.,* pp. 747-752, 1995.
59. A. Kosushi and K. Yamafuji, "Development and motion control of the all direction steering-type robot (1st report: a concept of spherical shaped robot, roll and running control)," *Proceedings of 9th Japanese Robotics Conference,* Japan, 1991.
60. E. P. Krotkov, R. G. Simmons, and W. L. Whittaker, "Ambler: Performance of A Six-legged Planetary Rover," *Acta Astronautica,* vol. 35, no. 1, 1995.
61. J. P. LaSalle, "Stability theory for difference equations," *Studies in Ordinary Differential Equations*, MAA Studies in Math., Amer. Math. Assoc., pp.1-31, 1977.
62. K. K. Lee and Y. Xu, "Human sensation modeling in virtual environments," *Proceedings of 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol.1, pp. 151-156, 2000.
63. T. C. Lee, K. T. Song, C. H. Lee, and C. C. Teng, "Tracking control of unicycle-modeled mobile robots using a saturation feedback controller," *IEEE Transactions on Control System Technology,* vol. 9, no. 2, pp. 305-318, 2001.
64. S. Lee and H. S. Lee, "An advanced teleoperator control system: design and evaluation," *Proc. of 1992 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 859-864, 1992.
65. A. U. Levin and K. S. Narendra, "Control of nonlinear dynamical systems using neural networks: controllability and stabilization," *IEEE Transactions on Neural Networks,* vol. 4, no. 2, pp. 192-206, 1993.
66. A. U. Levin, "An analysis method for estimating the domain of attraction for polynomial differential equations," *IEEE Transactions on Automatic Control,* vol. 36, pp. 2471-2475, 1994.
67. R. A. Lindemann and H. J. Eisen, "Mobility analysis, simulation and scale model testing for the design of wheeled planetary rovers", *Missions, Technologies, and Design of Planetary Mobile Vehicles,* 1992.
68. H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, Norwell, MA: Kluwer Academic Publishers, 1998.
69. H. A. Malki and A. Moghaddamjoo, "Using the Karhunen-Loe transformation in the back-propagation traning algorithm," *IEEE Trans. on Neural Networks,* vol. 2, no. 1, pp. 162-165, 1991.
70. T. McGeer, "Passive dynamic walking", *International Journal of Robotics Research,* 1989.

71. W. T. Miller, R. S. Sutton, and P. I. Werbos, Eds., *Neural Networks for Control*, Cambridge, MA: MIT press, 1990.
72. E. G. John , *Modern Mechanix and Inventions*, June, 1935.
73. J. F. Montgomery and G. A. Bekey, "Learning helicopter control through "teaching by showing," *Proc. IEEE Int. Conf. on Decision and Control*, vol. 4, pp. 3647 -3652, 1998.
74. G. C. Nandy and Y. Xu, "Dynamic Model of A Gyroscopic Wheel," *Proc. of the 1998 IEEE International Conference on Robotics and Automation* , vol. 3, pp. 2683-2688, 1998
75. R. Nakajima, T. Tsubouchi, S. Yuta, and E. Koyanagi, "A development of a new mechanism of an autonomous unicycle," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems,* vol. 4, pp. 3658-3663, 1997.
76. M. Nechyba, "Learning and validation of human control strategy," Ph.D. Thesis, Carnegie Melon University, 1998.
77. M. Nechyba and Y. Xu, "Stochastic similarity for validating human control strategy models", *IEEE Transactions on Robotics and Automation*, vol. 14 , no. 3, pp. 437-451, June, 1998.
78. M. Nechyba and Y. Xu, "Cascade neural networks with node-decoupled extended Kalman filtering," *Proc. IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, vol. 1, pp. 214-219, 1997.
79. S. H. Oliver and D. H. Berkebile, *Wheels and Wheeling,* Smithsonian Institution Press, 1974.
80. G. Oriolo and Y. Nakamura, "Control of mechanical systems with second-order nonholonomic constraints: underactuated manipulators," *Proc. of the 30th IEEE Conf. on Decision and Control*, vol. 3 pp. 2398-2403, 1991.
81. Y. Ou and Y. Xu, "Balance control of a single wheel robot," *Proc. of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems* , vol. 2, pp. 1361-1366, 2002.
82. A. Palmer, *Riding High: the Story of the Bicycle,* Dutton, NY, 1956.
83. S. J. Raudys and A. K. Jain, "Small sample size effects in statistical pattern recognition: recommendations for practitioners," *IEEE Transactons PAMI-13*, no. 3, pp. 252-264, 1991.
84. S. J. Raudys and A. K. Jain, "Small sample size problems in designing artificial neural networks," in I. K. Sethi and A. K. Jain, Eds. *Artificial Neural Networks and Statistical Pattern Recognition Old and new Connections*, Elsevier Science Publishers, pp. 35-50, 1991.
85. M. F. Redondon and C. H. Espinosa, "Neural networks input selection by using the training set," *International Joint Conference on Neural Networks*, vol. 2, pp. 1189 -1194, 1999.
86. M. Reyhanoglu, A.J. van der Schaft, N.H. McClamroch, and I. Kolmanovsky, "Dynamics and control of a class of underactuated mechanical systems," *IEEE Transactions on Automatic Control,* vol. 44, no. 9, pp. 1663-1671, 1999.
87. C. Rui and N. H. McClamroch, "Stabilization and asymptotic path tracking of a rolling disk," *Proc IEEE Int. Conf. on Decision and Control,* vol 4, pp. 4294-4299, 1995.
88. F. Saito, T. Fukuda, and F. Arai, "Swing and locomotion control for two-link brachiation robot," *IEEE Control Systems Magazine,* vol.14, no.1, pp. 5-12, Feb., 1994.
89. C. Samson, "Time-varying feedback stabilization of car like wheeled mobile robot," *Int. Journal of Robotics Research,* vol. 12, no. 1, pp. 55-64, 1993.

90. C. Samprit and S. H. Ali, *Sensitivity Analysis in Linear Regression*, New York: John Wiley & Sons, 1988.
91. N. Sarkar, "Control of vehicles with two steerable wheels," *ICRA'97 Workshop Innovative Design of Wheeled Mobile Robots*, 1997.
92. S. Schael and C. G. Atkeson, "Robot juggling: implementation of memory-based learning", *IEEE Control Systems Magazine,* vol.14, no.1, pp.57-71, Feb., 1994.
93. G. A. Seber and R. B. Schnabel, *Nonlinear Regression*. New York: Wiley, 1989.
94. P. Seibert, "Stability under perturbations in generalized dynamical systems," in J. P. LaSalle and S. Lefschetz Editors, *Nonlinear Differencial Equations and Nonlinear Mechanics*, New York: Academic Press, pp.1-31, 1977.
95. D. Shevitz and B. Paden, "Lyapunov stability theory of nonsmooth systems," *IEEE Transactions on Automatic Control*, vol. 39, no. 9, pp. 1910-1914, 1994.
96. Z. Sheng and K. Yamafuji, "Postural stability of a human riding a unicycle and its emulation by a robot," *IEEE Transactions on robotics and automation,* vol. 13, no. 5, pp. 709-20, 1997.
97. R. C. Simpson and S. P. Levine, "Adaptive shared control of a smart wheelchair operated by voice control," *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'97*, vol. 2, pp. 622-626, 1997.
98. A. Smola, "General cost function for support vector regression," *Proceedings of the Ninth Australian Conf. on Neural Networks,* pp. 79-83, 1998.
99. O. J. Sordalen C. and Canudas de Witt, "Exponential control law for a mobile robot: extension to path following," *Proc. IEEE Int. Conf. on Robotics and Automation,* vol. 3, pp. 2158-2163, 1992.
100. M. W. Spong, "The swing up control problem for the acrobot," *IEEE Control Systems Magazine*, vol.15, no.1, pp.49-55, Feb., 1995.
101. J. M. Stepper, K. W. Baur Jr., and S. K. Rogers, "Integrated feature and architecture selection," *IEEE Transactons on Neural Networks*, vol. 7, no. 4, pp. 1007-1014, 1996.
102. T. O. Summers, "Gyro stabilized vehicle," U.S. Patent, #3,410,357, 1968.
103. K. A. Tahboub and H. H. Asada, "A semi-autonomous control architecture applied to robotic wheelchairs," *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1999. IROS '99.*, vol. 2, pp. 906-911, 1999.
104. A. Tayebi and A. Rachid, "A unified discontinuous state feedback controller for the path-following and the point-stabilization problems of a unicycle-like mobile robot," *Proc. IEEE Int. Conf. on Control Applications*, pp. 31-35, 1997.
105. R. Thawonmas and S. Abe, "Feature reduction based on analysis of fuzzy regions," *Proc. 1995 International Conference on Artificial Neural Networks*, vol. 4, pp. 2130-2133, 1995.
106. B. Tibken and K. F. Dilaver, "Computation of subsets of the domain of attraction for polynomial systems," *Proc. IEEE Int. Conf. on Decision and Control*, pp. 2651-2656, 2002.
107. S. Tsai, E. D. Ferreira, and C. J. J. Paredis, "Control of the Gyrover," *Proc. of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems* , pp. 179-184, 1999
108. V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.

109. D. W. Vos and A. H. Von Flotow, "Dynamics and nonlinear adaptive control of an autonomous unicycle: theory and experiment," *Proceedings of the 29th IEEE Conference on Decision and Control,* 1990.
110. K. Waldron, *The Adaptive Suspension Vehicle*, MIT Press, 1989.
111. I. A. Weaver, "Gyroscopic vehicle steering stabilizer," U.S. Patent, #1,945,874, 1934.
112. Y. Xu, H. B. Brown, and K. W. Au, "Dynamics mobility with single-wheel configuration," *Int. J. of Rob. Res.,* vol. 18, no. 7, pp. 728-738, 1999.
113. Y. S. Xu and L. W. Sun, "Stabilization of a gyroscopically stabilized robot on an inclined plane," *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3549-54, 2000.
114. Y. S. Xu and L. W. Sun, "Dynamics of a rolling disk and a single wheel robot on an inclined plane," *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 811-816, 2000.
115. Y. Xu, K. W. Au, G. C. Nandy, and H. B. Brown, "Analysis of actuation and dynamic balancing for a single wheel robot," *Proc. IEEE/ISJ Int. Conf. on Intelligent Robots and Systems,* vol.4, pp.3658-3663,1998.
116. Y. Xu, W. Yu, and K. Au, "Modeling human control strategy in a dynamically stabilized robot," *Proc. of the 1999 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 2, pp. 507-512, 1999.
117. X. Yun, "Rigid body motion analysis towards rotary vehicle", *ICRA'97 Workshop Innovative Design of Wheeled Mobile Robots*, 1997.
118. J. Yang, Y. Xu, and C. S. Chen, "Human action learning via hidden Markov model," *IEEE Transactons on System, Man, and Cybernetics*, vol. 27, no. 1, pp. 34-44, 1997.
119. J. Yang, Y. Xu, and C. S. Chen, "Hidden Markov model approach to skill learning and its application to telerobotics," *IEEE Transactions on Robotic and Automation,* vol. 10, no. 5, pp. 621-631, 1994.
120. Y. Yokokohji, A. Ogawa, H. Hasunuma, and T. Yoshikawa, "Operation modes for cooperating with autonomous functions in intelligent teleoperation systems," *Proc. of the 1993 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 510-515, 1993.
121. S. You, T. Wang, J. Wei, F. Yang, and Q. Zhang, "Share control in intelligent arm/hand teleoperated system," *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2489-94, 1999.
122. S. Yuta, "A novel mechanism of an autonomous unicycle," *ICRA'97 Workshop Innovative Design of Wheeled Mobile Robots*, 1997.
123. J. Zabczyk, "Some comments on stabilizability," *Applied Math. Optimization*, vol. 19, pp. 1-9, 1989.
124. K. W. Au and Y. Xu, "Decoupled dynamics and stabilization of single wheel robot," *Proc. IEEE/RSJ Int. Workshop on Intelligent Robots and Systems,* vol. 1, pp. 197-203, 1999.

Index