Mobile Assistive Robot with Shared Control

by

Huanran Wang, M.Sc.

A Dissertation submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of

Doctor of Philosophy
in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
April, 2014
Abstract

The rapidly aging population and the existence of a large number of people with different kinds of disabilities pose a great challenge to engineers and researchers to design and build suitable and practical healthcare robotic systems. There are many applications and potential for assistive robotic systems in the area of healthcare. In this thesis, a prototype of a novel mobile assistive robot for senior and disabled people has been developed to help the group transport heavy objects in their daily life.

Several technical problems related to the design of the system are addressed and solved. In this robot prototype, a haptic device is used as the input interface to build the haptic interaction channel and compensate for the loss of visual or auditory capabilities in the target user. A nonlinear tracking controller is designed for the robot to follow the user. An obstacle avoidance controller is developed based on the deformable virtual zone (DVZ) principle, in order for the robot to avoid obstacles. A shared control problem is also studied in this thesis. Since the user may not have a normal level of mobility and cognitive capabilities, a stable and optimal adaptive servo-level shared control algorithm is applied to properly allocate the input authority between the human and the artificial intelligence when the robot encounters obstacles. Extensive experiments have been conducted to validate the control algorithms developed. The results demonstrate that in a cluster environment, the proposed robot can smoothly track the user.
Acknowledgments

I would like to express my special appreciation and thanks to my advisor Professor Dr. Xiaoping P. Liu, you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a PhD. Your advice on both research as well as on my career have been priceless. I would also like to thank my committee members, Professor Chunsheng Yang, Professor Abdulmotaleb El Saddik, Professor Mojtaba Ahmadi, and Professor Samuel A. Ajila for serving as my committee members even at hardship. I also want to thank you for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions, thanks to you. All of you have been there to support me when I recruited patients and collected data for my Ph.D. thesis.

A special thanks to my family. Words cannot express how grateful I am to my mother and father for all of the sacrifices that you’ve made on my behalf. Your support for me was what sustained me thus far. I would also like to thank all of my friends who supported me in writing, and incented me to strive towards my goal.
Table of Contents

Abstract ii

Acknowledgments iii

Table of Contents iv

List of Tables ix

List of Figures x

1 Introduction 1

1.1 Motivations .................................................. 1

1.2 Literature Review .............................................. 4

1.2.1 Review of Assistive Robotic Systems ................. 4

1.2.2 Review of Shared Control Algorithms ............... 8

1.2.3 Review of Haptic Interaction and Haptic Shared Control ... 11

1.3 Problem Statement ............................................. 12

1.4 Objectives of the Thesis and Summary of Results ......... 15

1.5 Application Scenario of the Proposed Mobile Assistive Robot . . . . 16

1.6 Thesis Contributions ........................................... 17

1.7 Outline of the Thesis .......................................... 19
# Overview of the Proposed Robot

2.1 Introduction ................................................................. 21
2.2 Design Purposes of the Mobile Assistive Robot ...................... 21
2.3 System Overview ............................................................. 22
   2.3.1 Mobile Robot .......................................................... 23
   2.3.2 Haptic Interface ...................................................... 25
   2.3.3 Connection of the Mobile Robot and Haptic Interface .......... 25
2.4 System Software ............................................................. 26
   2.4.1 Advanced Robotics Interface for Applications (ARIA) API for Mobile Robot .................................................. 26
   2.4.2 OpenHaptic for Haptic Device (Phantom Omni) ................. 27
   2.4.3 System Software Architecture .................................... 27

# Mobile Robot Controller Design

3.1 Introduction ................................................................. 30
3.2 Mathematical Models of the Mobile Robot ................................ 31
   3.2.1 Kinematics Model of the Nonholonomic Mobile Robot .......... 31
   3.2.2 Dynamic Model of the Nonholonomic Mobile Robot ............ 32
3.3 The Mobile Robot Controller Design .................................... 35
   3.3.1 Controller based on Rectangular Coordinate .................... 36
   3.3.2 Controller based on Polar Coordinate ............................ 40
3.4 Adaptive Windowing Velocity Estimation Based on Fuzzy Logic .... 44
   3.4.1 Fuzzification ......................................................... 46
   3.4.2 Fuzzy Rules .......................................................... 46
   3.4.3 Defuzzification ...................................................... 48
   3.4.4 Velocity Calculation ................................................. 48
List of Tables

3.1 Fuzzy Rules ...................................................... 48
List of Figures

1.1 The application scenario of the proposed mobile assistive robot . . . . 17
1.2 The Organization diagram of the thesis . . . . . . . . . . . . . . . . . . . . . 19
2.1 Overview of the developed assistant system . . . . . . . . . . . . . . . . . . . 23
2.2 Overview of the PeopleBot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
2.3 Overview of Phantom Omni Device . . . . . . . . . . . . . . . . . . . . . . . 25
2.4 The connection diagram of the mobile assistive robot . . . . . . . . . . . 26
2.5 System software architecture . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
3.1 Kinematics model of the nonholonomic mobile robot . . . . . . . . . . 32
3.2 Dynamic model of the nonholonomic mobile robot . . . . . . . . . . . . 33
3.3 Rectangular coordinate of the Mobile assistive robot . . . . . . . . . . . 37
3.4 Polar coordinate of the Mobile assistive robot . . . . . . . . . . . . . . . . . 41
3.5 Diagram of fuzzy logic windowing velocity estimation method . . . . . 45
3.6 Fuzzification membership function . . . . . . . . . . . . . . . . . . . . . . . 46
3.7 The angle limit of the Phantom Omni . . . . . . . . . . . . . . . . . . . . 50
3.8 The feedback force on the Phantom Omni . . . . . . . . . . . . . . . . . . 53
4.1 DVZ principle diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
4.2 The DVZ shape diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59
5.1 Adaptive servo level shared control framework . . . . . . . . . . . . . . . . 66
5.2 The difference between convexity and concavity . . . . . . . . . . . . . . 68
5.3 Diagram of the stability analysis for shared control: $U_h \subset U_r$  

5.4 Diagram of the stability analysis for shared control: $U_r \subset U_h$  

5.5 Diagram of the stability analysis for shared control: $U_r \cap U_h \neq \emptyset$, $U_r \not\subset U_h$ and $U_h \not\subset U_r$.  

5.6 Diagram of the stability analysis for shared control: $U_r \cap U_h = \emptyset$.  

6.1 Illustration of the equal interval method  

6.2 Illustration of the haptic feedback force $f_{va}$ for VAHSC.  

7.1 The designed trajectory experiment for the robot with the rectangular coordinate controller ($45^\circ$).  

7.2 The designed trajectory experiment for the robot with the rectangular coordinate controller ($90^\circ$).  

7.3 Human input experiment with the rectangular coordinate controller.  

7.4 The feedback force in the human input experiment with the rectangular coordinate controller.  

7.5 The changing window size of the proposed velocity estimation method.  

7.6 The designed trajectory experiment for the robot with the polar coordinate controller ($45^\circ$).  

7.7 The designed trajectory experiment for the robot with the polar coordinate controller ($90^\circ$).  

7.8 Human input experiment with the polar coordinate controller.  

7.9 The feedback force in the free space moving.  

7.10 The changing window size of the proposed velocity estimation method.  

7.11 Robot moving without safety control algorithm.  

7.12 $\phi_e$ and rotational velocity (no safety control algorithm).  

7.13 Robot moving with safety control algorithm.  

7.14 $\phi_e$ and rotational velocity (safety control algorithm).
7.15 Comparison experiment between the two shared controllers
(a) comparison between robot trajectories (b) comparison between
the value of the objective functions .................................. 114
7.16 Experimental environment ........................................... 115
7.17 Wall following experiments (a) human and robot trajectory (b) $\alpha_s$ in
the experiment .......................................................... 118
7.18 Narrow corridor passing experiment (door width: 880mm) (a) human
and robot trajectory (b) $\alpha_s$ in the experiment ......................... 120
7.19 Narrow corridor passing experiment (door width: 680mm) (a) human
and robot trajectory (b) $\alpha_s$ in the experiment ......................... 121
7.20 Door passing experiments (a) human and robot trajectory (b) $\alpha_s$ in the
experiment .......................................................... 122
7.21 Experimental environment ........................................... 123
7.22 Complex environment experiment without haptic feedback (a) human
and robot trajectory (b) $\alpha_s$ in the experiment ......................... 124
7.23 Complex environment experiment with haptic feedback (a) human and
robot trajectory (b) $\alpha_s$ in the experiment ......................... 125
7.24 Comparison of the allocation weights in different shared controllers
1. without haptic feedback 2. with haptic feedback .................... 126
Chapter 1

Introduction

1.1 Motivations

According to a study conducted by the Canadian government in 2006, people over 65 years of age represent 13% of the population [1]. By 2026, one Canadian in five will have reached age 65 [2]. As people grow old, their mobility and cognitive level will decline dramatically. This decline will make it difficult for them to perform basic daily activities. For example, many senior people do not have enough strength to move heavy objects or to move themselves from one place to another. Therefore, they need help accomplishing their daily activities. For instance, when senior people are walking, they might need help from others to prevent them from falling. When they are shopping in a grocery store, they often need someone to carry their groceries.

In addition to the aging population, we have a group of people who have different levels of disabilities. They may have a visual impairment, a hearing impairment or even have lost their mobility. Disabled people with various types of disabilities also have specific needs in their daily life. For example, blind people need someone to guide them. Deaf people need someone to translate their sign language. A universal solution is not yet available to help these people cope with their specific needs. These
CHAPTER 1. INTRODUCTION

two groups of people need specialized assistance and medical attention to live a normal life.

The current practice is to move these two groups of people into a centralized healthcare institution. Medical experts formulate a series of treatments for these individuals to help them maintain their mobility level. However, given the increasing number of seniors and the shortage of experts and therapists, the cost of healthcare will dramatically increase with the aging population. This will be a heavy burden on the healthcare system. Most importantly, these people cannot live an independent life, which is an important factor of the quality of modern life. The around-the-clock care from others can surely facilitate the life of these people, but the lack of adequate exercise and independence may influence their physical and mental health. For example, if senior peoples daily activities are performed by others with the goal helping them, their muscles will not get enough exercise and will gradually lose their functionality. This muscle loss will increase the aging process.

Helping seniors and disabled people with their daily needs to ensure that they can lead an independent life is an urgent problem that must be solved within a reasonable cost range. Robotic technology provides many good solutions to this problem. It is well known that the merits of a robotic system are repeatability, high precision movements, and high load capacity. These merits compliment well the time and labor intensive work of assisting the target population. The assistive robot is a category of healthcare robots that have been widely used in the medical treatment and daily life of senior and disabled people. The assistive robot is a subcategory of assistive technology, a field that aims to improve independent living and the quality of life of people who suffer from chronic illnesses or loss of sensory, motor, or cognitive abilities. The recent development of electrical devices such as sensors, actuators, and input devices, together with the high level artificial intelligence of these robots, have
created a very promising field of research for engineers and researchers looking to
design and build suitable and practical assistive robotic systems. A comprehensive
review can be found in the next section.

There are many practical and technical challenges in the development of assistive
robotic systems. On the practical side, there are many assistive tasks that have not
received attention from researchers and engineers. By now, certain levels of assistance
should be provided by assistive robots.

On the technical side, there are two important challenges that should be consid-
ered before designing assistive robots. The first technical challenge is to design a
stable adaptive shared control algorithm to enable the assistive robot system to work
cooperatively with the human user. Senior and disabled people have physical and
cognitive limitations. With these limitations, they may not always be able to gener-
ate the correct control signals for the robot. Therefore, a shared control algorithm
should compensate for this issue and help generate the correct control inputs for the
system. Shared control enables the user and the robot to share the common exe-
cutable platform and make the best use of human and artificial intelligence. For the
assistive robot, the human user and the robot intelligence share the same mechanical
platform to execute certain tasks. The assistive robotic systems must be designed
to work closely with the human user and be physically connected to the user. The
stability of the control algorithm is thus the first consideration in the design process.
An unstable and unsafe algorithm will damage the robot or even possibly injure the
user.

Developing a robot with a sustainable and robust interaction channel is the second
technical challenge. One problem with the human-robot interaction is the limited
sensory capabilities of certain senior people, which acts as a barrier for a stable
interaction. Vision and auditory sensing are the most important sensory inputs for
peoples everyday lives; for seniors, sensory and motor capabilities diminish with age. The changes in vision and auditory sensing are the most dramatic, when compared to the losses of the other senses [3]. The human-robot interaction needs a sustainable and robust interaction channel to exchange the information between the human user and the robot with a certain level of stability. It is not suitable to rely only on vision and auditory to build the stable human-robot interaction channel; another suitable channel should be added to complement the vision and auditory loss. It is known that haptic sense is reliable compared to those two senses, as the nerves for this sense are deep in the human muscles and are stronger than the vision and auditory sensing nerves. Therefore, in this thesis, the humans haptic channel is adopted as the interaction channel to compensate the limited sensory capability of the user.

1.2 Literature Review

In this section, we first present an overview of the different kinds of assistive robots. Then we introduce the shared control technique and different shared control algorithms. Last, we review some important literature on haptic interaction and haptic shared control.

1.2.1 Review of Assistive Robotic Systems

There is substantial research on the assistive robotic systems used for healthcare applications. From the current literature, there are three main categories of assistive robots: manipulation aids, mobility aids and cognitive aids.

Manipulation aids

Manipulation aids can be: fixed, portable, or mobile.
Fixed manipulation aids: These manipulation aids are usually composed of a lightweight stationary robot arm used as the manipulator, and of a computer used to interact with humans. The user commands the robot arm to bring, move, and place objects. One research prototype assistive robot, ProVAR, has been developed to help disabled office workers manipulate and operate devices [3]. The AfMASTER/RAID workstation, which has a similar concept, was developed in the European Union. This robot is placed close to the user’s own office space to help them with his or her daily work [4]. The researchers at Scuola Superiore Sant’Anna in Pisa developed the kitchen robot called Giving-A-Hand. It is a low degree-of-freedom device mounted on the front rail of a kitchen counter. The main function of this robot is to help people move food containers, set the cooking time on the cooking facilities, and open the doors cooking devices [5]. The Handy-1 robot was developed in the UK to help people with cerebral palsy. This robot has been successful commercially because of its concept and simplicity [6]. A recent study in Nature reports that two patients with tetraplegia are able to reach for and grasp objects using robotic arms that they control directly with their brain activity. A brain-computer interface (BCI) using a BrainGate neural interface system helps transfer their brain activity to the movement of the robot arm [7].

Portable and mobile manipulation aids: Portable and mobile manipulators have a similar approach: place the manipulator on a mobile base that is either a human controlled base or an autonomous moving base. MANUS, developed by ARM Exact Dynamics, is an integrated system whose robot arm is fixed on the wheelchair. The system is controlled using a joystick and a number pad on the wheelchair [8] [9]. Helpmate [10] is a very successful medical robot application that has been used by approximately 80 hospitals worldwide. As an autonomous transport system, this robot carries medical data records including X-ray photos, diagnostic samples, and
medical instruments. MovAid research robot platform, in Italy, adds manipulation and vision to the robot to help users manipulate objects and operate devices in home-like environments [11]. With its advanced navigation and sensing technique, the Care-O-bot, developed by Fraunhofer IPA, has been commercialized to be used as a physical support for the people who need the assistance [12]. A teleoperated mobile manipulator, Dusty, takes objects from the floor and brings them to a comfortable height for users [13]. It has a high success rate (98.4%) when autonomously grasping 25 objects considered important by people with amyotrophic lateral sclerosis. A home assistant robot developed at the University of Tokyo performs chores in peoples home while they are away [14]. This life-sized humanoid robot can perform various daily household tasks including kitchen tasks, clothes handling, and sweeping.

**Mobility aids**

There are three types of mobility aids: electric wheelchairs with autonomous navigation features, smart walkers, and exoskeletons.

**Wheelchair navigation systems:** The system is the add-on to the commercial wheelchair. Advanced sensors such as sonar, laser, radar, and camera, are mounted on the wheelchair. The NavChair [15] is an autonomous wheelchair system providing mobility to persons who cannot power the wheelchair due to motor, sensory, perceptual, or cognitive impairments. The NavChair contains three operating modes: general obstacle avoidance, door passage and automatic wall following. In the USA, Hephaestus is an accessory to the commercial wheelchair and can be integrated into a variety of wheelchair bands to upgrade the wheelchair to a smart wheelchair [16]. The Wheelseley [17] and Kares [18] use the vision information in their wheel chair system for the scene analysis and way-finding tasks. The Personal Mobility and Manipulation Appliance (PerMMA) developed at the University of Pittsburgh provides
people with severe physical disabilities enhanced assistance in both mobility and manipulation [19]. This robot integrates a powered wheelchair and two robotic arms to assist its users for their daily activities that require mobility and manipulation.

**Walking assistance systems:** This kind of robot is built on a walker. The robot cannot drive itself, but can actively steer and stop by collecting information from its sensors. A very good example is the Pam-Aid, which acts like a wheeled walker. Pam-Aid is equipped with different distance sensors (ultrasonic, laser, etc) to detect obstacles in front of the system. The brakes in the system were used to hold back the user from colliding with those obstacles [20]. GaitEnable built by Ahmadi and his team at Carleton University is used for gait rehabilitation [21]. The system has two main components: a patented mobile system (a bungee mobility trainer) and a reactive control system. This system has been successfully commercialized and applied to help many patients.

**Exoskeletons:** An exoskeleton is the external skeleton that supports the human body and amplifies human power. Exoskeletons were initially developed by the US Defense Department to increase the lifting and carrying capabilities of their soldiers. Later on, researchers and engineers extended their application to the medical field. Bionik Laboratories develops assistive devices targeting people with diseases and injuries that impact their mobility. Exolegs is one of the products that helps the patient stabilize while standing and walking [22]. Ekso, developed by eksobionics, is a wearable exoskeletons suit. This suit enables individuals with lower extremity weakness to stand and walk with a natural, full weight bearing gait [23].

**Cognitive aids**

The cognitive aid robots focus mainly on assistive tasks for people with mental health issues. This kind of robot is developed as a small pet or toy-like robot that interacts
with patients. For example, a baby harp seal robot, PARO, developed in Japan, is
designed as a therapeutic device for senior people. To interact with people, this seal
robot can give light, sound, and tilt when its body is touched [24]. The CosmoBot de-
developed by AnthroTronix, Inc is used to stimulate the speech and mobility of children
with cerebral palsy [25]. In the ROBADOM project, a robot is designed as a butler
to help researchers study the social context of designing social robots and examine
the perception of the expressivity of the robot [26].

1.2.2 Review of Shared Control Algorithms

The concept of shared control is used extensively in human robot interaction systems.
From teleoperation systems and medical training systems to assistive robotic systems
and more, shared control is adopted to solve a variety of problems [27]. There are
two general categories for the shared control: task level shared control and servo level
shared control.

The task level shared control is the macro level control. The human user just
gives a general order to the robot to complete a specific task such as passing the door,
moving along the hallway, avoiding the obstacles, etc. The robot will specifically carry
out this order. The macro level shared control can be found in the teleoperation
application [28–32]. The user has control of the mobile robot, but when a given
situation is triggered, such as the appearance of the obstacles, the robot takes control
to perform the obstacle avoidance action. In another situation, the user performs the
macro level movement, and when the robot moves into a certain area, it performs
the micro level movements to place itself on the target. In some assistive robot
applications [33–36], a basic set of movements, such as following the wall, avoiding
the obstacle or passing the doorway are recorded in the robot in advance. These
movements may be difficult for the target user to carry out. With these applications,
the user only has to manually select the type of movements needed to deal with the complex real situation. In other conventional autonomous robot systems [37–39], the user just gives the robot a destination. The robot then gets there by itself using autonomous planners.

From these macro level shared control applications, we can conclude that this kind of shared control is suitable for a situation where the user and the robot do not have close physical contact. It is not suitable for the system that has a physical human-robot interaction. Because the robot may be fully autonomous for some situations, the user may feel uncomfortable and find the robot out of control. For the proposed assistive robots, the user and the robot are physically connected by the haptic device. The user should therefore have some level of authority over the robot at all times. In addition, too many autonomous actions will adversely affect the physical and mental health of the target group.

The servo level shared control is the micro level control. The shared control signal is generated for every sample time. The control input for the robot platform is a combination of human and autonomous control inputs. At any given time, the user is involved in controlling the movement of the robot. Equation (1.1) shows this idea:

\[
    u_s = (1 - \alpha_s)u_h + \alpha_s u_r.
\]  

Here, \( u_s \) is the shared control input for the robot platform, \( u_h \) and \( u_r \) are the human and autonomous inputs respectively. \( \alpha_s \) is the allocation weight to adjust the proportion of \( u_h \) and \( u_r \) in \( u_s \).

The allocation weight \( \alpha_s \) can be a fixed value or a real-time changing value. Based on these two different kinds of allocation weights, the servo level shared control can be divided into two categories: fixed servo level shared control and adaptive servo
level shared control.

The fixed servo level shared control is often used in a situation with a prior knowledge of the two inputs. For example, this kind of shared control can be used in dual-user haptic training for medical surgery [40–43]. The trainer knows the level of the trainee and can use the allocation weight $\alpha_s$ to adjust the dominance when training different level trainees. The adaptive servo level shared control is often used in a situation where some inputs may bring unstable results, or in which the environment may change dramatically. For example, a large-scale manipulator used for gripping and lifting heavy objects was developed at Umea University in Sweden [44]. In this manipulator, $\alpha_s$ is given by the magnitude of the operator reference velocity, the position along the path, the distance to the path, and the designed path and the duration of the interaction. This adaptive weight gives the optimal proportion for the inputs to complete the manipulating tasks. In the assistive robots, the adaptive servo level shared control plays a very important role in ensuring a stable and safe operation of the robot platform. Yu [45] uses an adaptive law in which $\alpha_s$ is calculated on the performance indices history of the user. Urdiales [46] and Poncela [47] use the average of the performance indecies to calculate $\alpha_s$. Li [48] uses a minimax multi-objective optimization algorithm to calculate the allocation weight.

We notice that the variation of these shared control algorithms is based on the purpose of their developed system. For example, the personal aid in [45] is used to compensate for lack of mobility and to monitor the health of the elderly; the history of performance indices is therefore very important for this research. The smart wheelchairs in [46–48] are used to remove onerous maneuvers as well as possible dangers like collisions and falls. The combination of performance indices such as safety, smoothness, and direction are therefore all used to calculate $\alpha_s$. As a result, the development of the shared controller should vary based on its application.
1.2.3 Review of Haptic Interaction and Haptic Shared Control

Haptic comes from a Greek word “haptesthai”, which means “of or relating to the sense of touch”. Humans use haptic to sense the weight and texture of objects in the world and to manipulate the objects that surround them objects [49]. There are two kinds of haptic sensations: kinesthetic sensations and tactile sensations. Kinesthetic sensations relate to the feeling of an objects motion. This sensation originates from the nerves in a human’s muscles, tendons, and joints. Tactile sensations relate to the feeling of the texture of the objects surface. This sensation originates from the nerves in a human’s skin. Since the application in this thesis is related to the motion of the objects, we only consider the kinesthetic sensation for the human user while he or she is operating the mobile assistive robot.

Haptic technology is a feedback technology that utilizes the haptic sensation by applying forces, vibrations, or motions to the human user. Haptic technology as an active research and engineering topic has been widely used in various applications such as surgery simulators [50] [51] [52], teleoperation [53] [54], biometrics authentication [55], rehabilitation [56] [57], human-robot interaction [58], etc.

In order to help the human understand the assistive robots intentions during a shared control process, haptic feedback has been introduced to the process, which is called haptic shared control (HSC). There are two general categories for HSC: fixed authority HSC (FAHSC) and variable authority HSC (VAHSC) [59].

In fixed authority HSC, the amount of haptic feedback is fixed during operation. This technique is used in the early stages of HSC. Rosenberg [60] uses FAHSC based on a virtual fixture to prevent the operator from exceeding the fixture in a teleoperation task. In his results, the operators performance increases by 70% when using
FAHSC. Abbink and Mulder investigate the level of authority problem for FAHSC, and give some important design guidelines [61]. FAHSC is suitable for teleoperation applications, in which the operator and the robot are far away from each other and do not have any physical contact. Therefore, the non-smooth characteristics of FAHSC will not affect the operators action. In current literature, the amount of the haptic feedback is given by a trial-and-error tuning approach. This simple approach gets good results but lacks mathematical analysis.

In variable authority HSC, the amount of haptic feedback changes based on some adaptive rules. In Abbink and Mulders work about a haptic gas pedal [62], the force feedback and stiffness change according to the distance between the operator and the leading vehicle. Goodrich develops a flight simulator [63] based on H-mode, which is an idea similar to VAHSC. The change of the authority is triggered by the pilots intention. VAHSC changes the authority smoothly during the operation of the system. Therefore, the operator feels more comfortable while the authority changes. Especially for certain physical human-robot interaction systems, with its smooth characteristics, VAHSC helps guarantee the safety operation of the system.

1.3 Problem Statement

While these robotic systems are very promising in assisting people to live a more independent life, there are several drawbacks in the current systems:

1. The interaction between the user and the system is not sufficient or not intuitive.
   Most of the current systems are meant to be autonomous. This reduces the burden on the user, which is of course a very desirable feature, but due to the low level of current artificial intelligence technology, most of the current systems cannot satisfy the various, different, and unstructured requirements of
CHAPTER 1. INTRODUCTION

an independent lifestyle. Moreover, due to the various mobility problems of the
user group, the input device should have a high degree of freedom input device
to track any input movement.

2. While the robotic systems meant for the above assistive tasks have been quite
intensively developed, there are some tasks, such as moving heavy objects (gro-
cery, furniture, etc.), that have received relatively little attention. For the people
whose upper limb power is too weak to manipulate heavy objects, the assistant
robot should act as a power amplifier and handle most of the weight. The re-
main ing weight is designated as an acceptable load for the user to bear as an
exercise.

3. The stability of the shared control algorithm is not well analyzed. In the pre-
vious work, different shared control algorithms are proposed based on various
design objectives. However, there is no rigorous stability analysis for these algo-
rithms, and little discussion of such analysis is found in the literature. Ren [64]
investigates the stability problem of his shared control problem using the control
Lyapunov function method. However, in his application, the human user and
the robot do not have any physical contact. As mentioned previously, shared
control algorithms vary from one another based on their applications. There-
fore, the stability analysis methods also vary slightly in different shared control
algorithms. However, the common problem in stability analysis for adaptive
servo-level shared control is to find a suitable range for the allocation weight to
guarantee the safety and stable operation of the system. To solve this common
problem, some fundamental framework should be investigated.

4. Whether or not there is an optimal solution for the shared control. We know
that for shared control, there is not only one controllers output that can satisfy
the user and the robots objective. These outputs are the solution for the shared control problem of the assistive robot. We therefore need an optimal solution to balance the human and the artificial intelligence of the system. This optimal solution needs to consider the users intentions and guarantee a safe operation of the system, at the same time.

5. Human communication channels are adequately used. In the literature, human-robot interaction is mostly through visual and auditory (voice command) channels. However, the elderly who need healthcare accommodations usually lose, partially or completely, their visual and/or auditory capabilities; thus, it may be challenging for them to interact with the assistive robot using voice, or to locate the robot visually. In addition, with age, the changes that occur to hearing and vision are the most dramatic when compared to the changes in the other senses [65]. An assistive robot that has the capability for haptic interactions may compensate well in this regard.

6. Whether haptic feedback will allow the assistive robot to better interact with human users in our proposed assistive robot. As discussed before, comparing to the other senses, haptic feedback is stable as age increases. For that reason, it is a good idea to add haptic sense to the assistive robot, to help users interact with it. Whether this feedback will improve the interaction between users and the assistive robot is still unknown. Since different assistive robots have different working objectives, we should perform experiments on our proposed assistive robot to see whether or not the haptic feedback is successful.
1.4 Objectives of the Thesis and Summary of Results

The first objective of this thesis is to design a mobile assistive robot with a haptic input device to help senior and disabled people manipulate heavy objects in an environment with obstacles. The haptic input device not only captures the users input and maps this input as the control signal for the robot, but it also sends feedback force for the user to interact with the robot through a haptic channel. Therefore, a tracking controller for the mobile robot should be developed to follow the users movements. In addition, a haptic rendering algorithm should also be developed to build the haptic interaction channel for the human-robot interaction.

In this thesis, substantial study and development has been done on the design and control of the mobile assistive robot. Two nonlinear tracking controllers have been developed to control the mobile robot. An adaptive windowing velocity estimation method based on fuzzy logic is used to solve the velocity estimation problem in the nonlinear controller. The safety control problem is solved by a revised constraints method. In order to generate the haptic feedback, the haptic rendering algorithm based on Hooks Law is implemented in the haptic device.

The second objective is to develop a stable and safe adaptive servo level shared control algorithm for our system. The users of the developed system are senior and disabled people who have limited sensory and mobility capabilities. These users may give the wrong control input to the robot, therefore the control algorithm should be able to correct this incorrect command and give the correct command to the robot. On the other hand, with the shared control algorithm, the human user can feel that during the whole operating process, they always have a certain level of control over the robot. This type of control reflects the same core idea as the assistive robot:
assistance is only provided when it is necessary. In addition, in order to optimize the usage of the human and artificial intelligences and find the best control input for the robot, we should consider the optimal output of the shared controller. By using this optimal control output, the robot could find the best way to both safely finish the task and respect the humans control.

In this thesis, a novel stable and safe adaptive servo level shared control algorithm specifically designed for our mobile assistive robot is developed. When the sensor on the robot detects obstacles, this algorithm will be activated. The two control inputs for the shared controller are generated by the tracking controller and the obstacle avoidance algorithm respectively. The obstacle avoidance algorithm is developed by the Deformable Virtual Zone (DVZ) principle. Based on a convex analysis, different situations in shared control are analyzed and the stable ranges of the shared controllers output are given. By using the multi-objective optimization concept, the shared control problem is modeled as a multi-objective optimization problem. We then use an equal interval method to solve this multi-objective optimization problem.

1.5 Application Scenario of the Proposed Mobile Assistive Robot

The developed mobile assistive robot can be used in grocery stores to help senior and disable people move heavy grocery items. This idea is illustrated in Fig. 1.1. The user holds the stylus of the phantom to control the mobile robots movements in the grocery store. The mobile robot holds the grocery items and moves based on the movements of the stylus. When the mobile robot encounters obstacles that are usually seen in the store, the robot will avoid the obstacle, whether or not the user
noted the obstacle.

1.6 Thesis Contributions

1. Developed a novel mobile assistive robot prototype for senior and disabled people to help them move heavy objects in a cluster environment. Since assistive robots are designed for specific assistive tasks, our proposed assistive robots are specially designed to help our target group of people in the grocery store.

2. Haptic feedback is applied to build a stable interaction channel in the human-robot interaction process. Since haptic feedback is very stable compared to the other senses, it is necessary to use this technique to compensate for the loss of other sensory due to the aging process.

3. Analyzed the stability of shared control. As a control system, the stable range of the controller is very important to ensure the stability of the system. For the assistive robotics system, the safety operation relies on the stability of the
robots control system. Therefore, to ensure the safe operation of the robot, we need to give the stable range of the robots controller. With the use of a convex analysis, the stable ranges of the shared controller in different situations are given. To the author’s best knowledge, the stability analysis of shared control receives little attention from current researchers. This is the first time the stable range of the shared controller is given to help design a safe and stable mobile assistive robot.

4. Used the multi-objective optimization technique to solve the optimal shared control problem. In the shared control process for an assistive robot, the inherent problem is to find a good balance between human and artificial intelligence, in order to accomplish the assigned tasks together. The shared controller should be able to achieve the objectives of both human and artificial intelligences. The best mathematical model for the shared control problem is the multi-objective optimization technique. In this thesis, we analyze different objectives for the assistive robot and utilize these objectives to model the specific shared control problems of the proposed mobile assistive robot, as a multi-objective optimization problem. Although in current literature some researchers talk about multi-objective optimization in shared control, they do not model the whole shared control problem as a multi-objective optimization problem. To the best of the author’s knowledge, this is the first time the shared control problem is modeled in an assistive robot, as a multi-objective optimization problem.
1.7 Outline of the Thesis

This thesis contains seven chapters. The organizational diagram of the thesis is shown in Fig. 1.2.

The introduction chapter gives the motivation of the research, the literature review, and the problem analysis. This chapter also outlines the thesis research and summarizes the contributions.

Chapter 2 introduces the overview of the developed mobile assistive robot system, as well as the different components of the systems hardware and software.

Chapter 3 presents the tracking control algorithm of the system. Two different tracking controllers for the mobile robot are developed. An adaptive fuzzy logic based windowing method is proposed to solve the velocity estimation problem in the controller. The safety control problem is solved by a revised constraints method. A haptic rendering algorithm is designed to generate haptic feedback based on the position error.
Chapter 4 presents the autonomous control of the system. The obstacle avoidance algorithm of the mobile robot is developed by the deformable virtual zone (DVZ) technique. A new calculation method for DVZ is also introduced.

Chapter 5 introduces the shared control function of the system. An adaptive servo level shared control algorithm is developed specifically for the proposed mobile assistive robot. A convex analysis is applied to analyze the stable range of the shared controller in different situations.

Chapter 6 introduces the multi-objective optimization of the shared control problem.

Chapter 7 presents the experimental results. The experiments are carried out to verify the effectiveness of the developed control algorithms.

Finally, Chapter 8 concludes the thesis and suggests some future work.
Chapter 2

Overview of the Proposed Robot

2.1 Introduction

In this chapter, we introduce the hardware platform and the software architecture of the mobile assistive robot. First, we analyze the design requirements of the proposed mobile assistive robot. Then we give the overview of the proposed mobile assistive robot. After this, different hardware components are introduced separately. Moreover, the software architecture is also introduced in this chapter.

2.2 Design Purposes of the Mobile Assistive Robot

In the last chapter, we proposed a novel mobile assistive robot to help seniors and disabled people move heavy objects in cluster environments such as grocery stores. From the users point of view, the design requirements are:

1. The user should be able to control the robots movements when moving and carrying heavy items.
2. The control interface should be easy to use. The user should be able to naturally and effortlessly use the controller.

3. The user should be able to feel the force feedback from the robot, in order to be informed of any potential dangers.

From the robot’s point of view, the design requirements for the robot are:

1. The robot should have a strong base to hold all heavy items.

2. The robot should be able to detect obstacles and avoid colliding with them.

3. The robot should be able to send force feedback to the user to inform them of any potential danger.

From the above requirements, we see that for the assistive robot, there are at least two hardware components needed. One is a control interface, which transfers the users control commands to the robot. This control interface should also have the haptic feedback feature, which can enable a haptic interaction between the user and the robot. The other hardware component is a moving platform that could carry heavy items and detect surrounding objects. This platform should be equipped with range sensors to detect obstacles.

### 2.3 System Overview

An overview of the proposed assistive robotic system is shown in Fig. 2.1. To achieve the design requirements, the system has two main hardware components: a mobile robot as the moving platform and a haptic device as the control interface. A flat wood panel is designed to fix the haptic device on the mobile robot and ensure there is no
movement between these two components. The basket holding the heavy objects is placed at the mobile robots base.

2.3.1 Mobile Robot

The mobile robot is a PeopleBot robot from MOBILEROBOTS, which is a suitable robotic platform for human-robot interaction (shown in Fig. 2.2). It is a nonholonomic mobile robot with two different types of wheels: two driving wheels and one omnidirectional wheel for balance [66]. The integrated high-resolution motion encoders and gyroscope provide data on the accurate position and orientation of the robot. The height of the mobile robot is 112cm, and it is suitable for the user to manipulate the haptic device. In addition to the motion components, many sensing
components are also integrated into the mobile robot. Range sensors, such as sonar sensors are placed on the base and top part of the robot. The base has 16 sonar sensors placed around the robot's body, and the top has 8 sonar sensors placed on the front side. One laser scanner is placed on the base to scan objects in front of the robot. Moreover, a Canon CCD camera is fixed on the top of the robot. In this thesis, we only use the sonar sensors around the base as range sensors. Because our application is in an environment that has randomized objects, these objects around the robot may become obstacles and influence its motions. The required range sensors should be able to sense the objects around the robot.
2.3.2 Haptic Interface

The haptic interface is a Phantom Omni (shown in Fig. 2.3) from Sensable Technology [67], which is lightweight and easy to mount to the top of the robot. The user holds and moves a stylus in 3D space in order to perform the haptic interaction. The Phantom Omni has 6 DOF positional sensing capabilities and 3 DOF haptic feedback capabilities. Because it is a 6 DOF robot arm, with torque transmitted through pre-tensioned cable, the manipulation of the device is quite smooth. It is very suitable for senior and disabled people to manipulate this haptic device.

2.3.3 Connection of the Mobile Robot and Haptic Interface

The portable notebook computer is used as a centre controller to execute commands from the input haptic device to the mobile robot. This notebook has a 1.4G Pentium-M CPU and 512 MB memory. Both the mobile robot and the haptic device are connected to the computer. The mobile robot is connected to the computer via a serial port. The haptic device is connected to the computer via an IEEE 1394 port.

Fig. 2.4 shows the connection diagram.
2.4 System Software

For the software part of the mobile assistive robot, we will first introduce the different software packages for the PeopleBot and the Phantom Omni haptic device. We will then present the software architecture.

Since the mobile robot and haptic device are commercial products, the suppliers provide appropriate software packages for developers to fully utilize these devices. These software packages are introduced in the following sections.

2.4.1 Advanced Robotics Interface for Applications (ARIA) API for Mobile Robot

This software package is provided by MobileRobots Inc. and can be used to control any of their models, e.g. AmigoBots, PeopleBots and Pioneers. ARIA stands
for ActivMedia Robotics Interface for Application and is an object-oriented Applications Programming Interface (API) developed in C++ and used for the creation of intelligent high-level client-side software [68]. Another important software used in this thesis is MobileSim, which is used to simulate MobileRobots platforms and their environments. Before executing any algorithm on the real robot, we test these algorithms on MobileSim and debug the program in order to avoid any dangerous movements from the mobile robot in the real application.

2.4.2 OpenHaptic for Haptic Device (Phantom Omni)

An OpenHaptics toolkit [69] is provided by SensAble Technology and can be used to develop the haptics/graphics applications with their haptic devices, e.g. Phantom Omni, Phantom Desktop and Phantom Premium. This toolkit developed in C++ provides classes of the QuickHaptics micro API. This API contains both of the haptic and graphic functions needed to build the haptic application in a virtual environment.

2.4.3 System Software Architecture

For the design of the software architecture, the different sample rates of the hardware should be considered. For the mobile robot, the sample rate of the hardware is around 100Hz. The range sensors, gyroscope, encoder of the motors, and tachometer update their information to the system in about 0.01 seconds. This is limited by the hardware configuration of the mobile robot. To ensure a reliable and stable haptic rendering algorithm, the sample rate of the encoders and motors of the haptic device should be 1000Hz. That is to say that the update speed should be a maximum of 0.001 seconds. Otherwise, the user will feel an unsmooth haptic interaction while operating the robot. To ensure the system runs on the different update speeds of the different
hardware components, a multi thread technique is applied to the software design. The algorithms related to the mobile robot run in the main thread. The haptic rendering algorithm runs in the secondary thread.

The system software architecture is shown in Fig. 2.5.

As can be seen in the figure, after initializing the Mobile robot and Phantom Device, the program is divided into two threads that run simultaneously. The mobile robot has a lower sample rate of around 100Hz. In order to ensure a reliable and stable haptic rendering algorithm, it should be placed into a separate, higher, sample
rate thread of 1000 Hz. The main thread (on the right) contains the shared control algorithm, which uses the human and autonomous control inputs to command the mobile robots movements. The secondary thread contains the haptic rendering algorithm, which utilizes the Phantoms information from the main thread. The thread creation and synchronization are completed by OpenHaptic. Visual Studio C++ 2005 is chosen as the tool to develop the above program.
Chapter 3

Mobile Robot Controller Design

3.1 Introduction

One of the objectives of the thesis is to develop a stable control algorithm for the mobile robot to track the users movements. This control algorithm must also provide the human control input for the shared control algorithm.

One possible approach is to use the Phantom as a joystick to control the movements of the robot. There is a linear relation between the position of the Phantoms stylus and the robots translational and rotational velocity. When a user moves the Phantoms stylus from its original position, the robot moves based on the difference in the stylus’s original and current positions. The larger the difference is, the faster the robot moves. This practical approach can control the robots movements, however, it cannot be mathematically proven that the controller of this approach can be stable all the time. Because the target group are senior and disabled people with limited sensibility and mobility, the fundamental requirement is to ensure a stable controller for the mobile robot to perform a safe human-robot interaction. Therefore, the design of a stable controller for the mobile robot is the primary objective of this thesis.

In this chapter, we first introduce the different mathematical models of the mobile
robot. We then choose a suitable model according to the specific features of the robot. Based on the chosen model, different tracking controllers are developed. To estimate the velocity of the Phantom used in the controller, we develop an adaptive windowing velocity estimation based on fuzzy logic. In addition, to ensure the safe operation of the assistive robot, we develop a safety control for its manipulation. Finally, a simple haptic rendering algorithm is developed, based on the tracking controller, to help the user interact with the robot.

3.2 Mathematical Models of the Mobile Robot

In the proposed system, the robot is a nonholonomic mobile robot, which restricts the condition that the translational velocity of the robot is always orthogonal to the driven wheels axis. There are two kinds of mathematical models for the nonholonomic mobile robot: the kinematics model and the dynamic model. These models are introduced below.

3.2.1 Kinematics Model of the Nonholonomic Mobile Robot

The coordinates of the mobile robot are shown in Fig.3.1.

As seen in the figure, the point \( P_m = [x_m, y_m]^T \) \((x_m, y_m \in \mathbb{R})\) is the reference position of the mobile robot in the world coordinate \( XY \). \( \theta_m \) \((\theta_m \in \mathbb{R}, -\pi < \theta_m < \pi)\) is the orientation of the mobile robot. If we assume the current translational and rotational velocity are \( v_m \) and \( \omega_m \) \((v_m, \omega_m \in \mathbb{R})\), respectively. The kinematics model can be expressed as:
CHAPTER 3. MOBILE ROBOT CONTROLLER DESIGN

3.2.2 Dynamic Model of the Nonholonomic Mobile Robot

The diagram for the dynamic model of the nonholonomic mobile robot is shown in Fig.3.2.

The nonholonomic mobile robot has an $n$-dimensional configuration subject to $m$...
constraints in the world coordinate \( XY \), and can be described by (detailed terms and definitions can be found in [70]):

\[
M(q)\ddot{q} + V_m(q, \dot{q})\dot{q} + F(\dot{q}) + G(q) + \tau_d = B(q)\tau - A^T(q)\lambda,
\]  

(3.3)

where \( M(q) \in R_{n \times n} \) is the inertia matrix for the mobile robot,

\[
M(q) = \begin{bmatrix}
m & 0 & md \sin \theta_m \\
0 & m & -md \cos \theta_m \\
md \sin \theta_m & -md \cos \theta_m & I
\end{bmatrix}
\]
$V_m(q, \dot{q}) \in \mathbb{R}^{n \times n}$ is the centripetal and coriolis matrix,

$$V_m(q, \dot{q}) = \begin{bmatrix}
    m d \dot{\theta}_m^2 \cos \theta_m \\
    m d \dot{\theta}_m^2 \sin \theta_m \\
    0
\end{bmatrix}$$

$F(\dot{q}) \in \mathbb{R}^{n \times 1}$ is the surface friction between the robot and the ground,

$G(q) \in \mathbb{R}^{n \times 1}$ is the gravitational vector,

$$G(q) = 0$$

$\tau_d \in \mathbb{R}^{n \times 1}$ is the unknown disturbances,

$B(q) \in \mathbb{R}^{n \times r}$ is the input transformation matrix,

$$B(q) = \frac{1}{r} \begin{bmatrix}
    \cos \theta_m & \cos \theta_m \\
    \sin \theta_m & \sin \theta_m \\
    R & -R
\end{bmatrix}$$

$\tau \in \mathbb{R}^{n \times 1}$ is the input torque for the motors on the mobile robot,

$$\tau = \begin{bmatrix}
    \tau_r \\
    \tau_l
\end{bmatrix}$$
CHAPTER 3. MOBILE ROBOT CONTROLLER DESIGN

\[ A(q) \in R_{m \times n} \text{ is the constraint matrix,} \]

\[ A^T(q) = \begin{bmatrix} -\sin \theta_m \\ \cos \theta_m \\ -d \end{bmatrix} \]

and \( \lambda \in R_{m \times 1} \) is the constraint forces.

\[ \lambda = -m(\dot{x}_m \cos \theta_m + \dot{y}_m \sin \theta_m)\dot{\theta}_m \]

The dynamic model provides much information about the mobile robot: position, velocity, acceleration, inertia, friction, etc. This information is very useful to build a stable and smooth controller. Unfortunately, it is very hard to utilize the dynamic model of Peoplebot to design the controller. From (3.3), we know that the control input of the mobile robot is the torque of the motor. However, we cannot access the torque information of the Peoplebot. As Peoplebot is a commercial robot with an integrated low level motion controller and an ARIA software package, it does not provide the torque information needed. ARIA only provides the right to read and write the translational and rotational velocity \( v_m \) and \( \omega_m \). Therefore, the kinematic model is chosen to design the controller for the Peoplebot.

3.3 The Mobile Robot Controller Design

The form of the kinematics model is simple, but it is a highly nonlinear model, since the first two equations contain trigonometric functions of the state variables. Consequently, the controller design is carried out by Lyapunov, a stable theory that is
suitable for Figure 3.3: Rectangular coordinate of the mobile assistive robot nonlinear systems. There are two kinds of coordinates for the kinematics model of the mobile robot. One is a rectangular coordinate, and the other is the polar coordinate. Based on the Lyapunov stable theory, we designed two kinds of controllers for the mobile robot. One controller is based on the rectangular coordinate system, while the other is based on the polar coordinate system. A performance comparison of these two controllers is shown in Chapter 7. Based on the experimental results, we chose the most appropriate one for the tracking controller of our mobile assistive robot. Here we introduce these two kinds of tracking controllers.

3.3.1 Controller based on Rectangular Coordinate

The rectangular coordinate arrangement of the system is shown in Fig.3.3. The points $P_h = [x_h, y_h]^T$ and $P_m = [x_m, y_m]^T$ are the reference positions of the human and of the mobile robot respectively, in the world coordinate $XY$.

Consider the following kinematics model for the nonholonomic mobile robot:

$$
\dot{P}_m = \begin{bmatrix}
\dot{x}_m \\
\dot{y}_m \\
\dot{\theta}_m
\end{bmatrix} = \begin{bmatrix}
\cos \theta_m & 0 \\
\sin \theta_m & 0 \\
0 & 1
\end{bmatrix} \begin{bmatrix}
v_m \\
\omega_m
\end{bmatrix}, \quad (3.4)
$$

where $P_m$ is the position and posture vector of the mobile robot. The cartesian coordinates for the robot are $x_m$ and $y_m$ respectively. The angle between the direction in which the robot is heading and the $X$ axis is $\theta_m$. The translational velocity is $v_m$, and the rotational velocity is $\omega_m$.

We define the error $P_e$ as:
Here, $x_h$ and $y_h$ represent the position of the Phantom stylus in the world coordinate. The tracking control problem can be defined as: With the inputs ($P_h = [x_h, y_h]^T$ and $P_m = [x_m, y_m]^T$) and the states ($[\dot{x}_h, \dot{y}_h, \dot{x}_m, \dot{y}_m]^T$), we want to make the mobile robot track the human users position.

$\alpha_{VI}$ is a virtual angle compare to the real angle $\theta_m$. If $\theta_e = 0$, $\alpha_{VI} = \theta_m$. Now, we assume the velocity of the mobile robot is
\[
\begin{bmatrix}
\dot{x}_m \\
\dot{y}_m
\end{bmatrix} = \begin{bmatrix}
v_m \cos \alpha_{VI} \\
v_m \sin \alpha_{VI}
\end{bmatrix}.
\] (3.6)

Considering the following candidate Lyaponuv function:

\[
V_1 = \frac{1}{2} x_e^2 + \frac{1}{2} y_e^2.
\] (3.7)

The derivative of (3.7) is

\[
\dot{V}_1 = x_e \ddot{x}_e + y_e \ddot{y}_e \\
= x_e (\dot{x}_h - \dot{x}_m) + y_e (\dot{y}_h - \dot{y}_m) \\
= x_e (\dot{x}_h - v_m \cos \alpha_{VI}) + y_e (\dot{y}_h - v_m \sin \alpha_{VI}).
\] (3.8)

If we make

\[
\begin{bmatrix}
v_m \cos \alpha_{VI} \\
v_m \sin \alpha_{VI}
\end{bmatrix} = \begin{bmatrix}
\dot{x}_h + c_x x_e + c_{VF} x_{VF} \\
\dot{y}_h + c_y y_e + c_{VF} y_{VF}
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\dot{x}_h + c_x x_e + \text{sign}(x_e) c_{VF} \sqrt{x_e^2 + y_e^2} \\
\dot{y}_h + c_y y_e + \text{sign}(y_e) c_{VF} \sqrt{x_e^2 + y_e^2}
\end{bmatrix}.
\] (3.9)

Here, we introduce a virtual force \( x_{VF} \) (on the X axis) and \( y_{VF} \) (on the Y axis), which acts like as a force that pushes the mobile robot to move into the direction which can minimizes the errors. This force utilizes the errors from both the X and Y axis. \( c_x, c_y \) and \( c_{VF} \) are the gains for the errors \( x_e, y_e \) and \( x_{VF} \) respectively.
By putting (3.9) into (3.8), we get

\[
\dot{V}_1 = x_e (\dot{x}_h - v_m \cos \alpha_V I) + y_e (\dot{y}_h - v_m \sin \alpha_V I)
\]

\[
= x_e \left( -c_x x_e - \text{sign}(x_e)c_{VF} \sqrt{x_e^2 + y_e^2} \right) + y_e \left( -c_y y_e - \text{sign}(y_e)c_{VF} \sqrt{x_e^2 + y_e^2} \right)
\]

\[
= -c_x x_e^2 - \text{sign}(x_e)x_e c_{VF} \sqrt{x_e^2 + y_e^2} - c_y y_e^2 - \text{sign}(y_e)y_e c_{VF} \sqrt{x_e^2 + y_e^2} < 0.
\]

(3.10)

For the rotational error, we have

\[
\theta_e = \alpha_V I - \theta_m.
\]

(3.11)

We then choose the composite candidate Lyapunov function as

\[
V = V_1 + \frac{1}{2} \theta_e^2.
\]

(3.12)

By putting (3.11) into (3.12) we get

\[
\dot{V} = \dot{V}_1 + \theta_e \dot{\theta}_e
\]

\[
= \dot{V}_1 + \theta_e (\alpha_V I - \dot{\theta}_m)
\]

\[
= \dot{V}_1 + \theta_e (\dot{\alpha}_V I - \omega_m).
\]

(3.13)

We can then choose the rotational controller as

\[
\omega_m = \dot{\alpha}_V I + c_\theta \theta_e,
\]

(3.14)

By substituting (3.14) into (3.13), we get
\[
\dot{V} = \dot{V}_1 + \theta_e (\dot{\alpha}_v - \omega_m)
\]
\[
= \dot{V}_1 - c_\theta \theta_e^2 < 0.
\] (3.15)

The derivation of the composite Lyaponov function is smaller than 0. Based on the Lyapunov stability theory, the system is therefore stable.

From equation (3.9), we can get the controller for the translational velocity of the mobile robot. Together with the rotational controller (3.14), we have the controller for the mobile robot

\[
\begin{aligned}
v_m &= \sqrt{\left(\dot{x}_h + c_x x_e + x_{VF}\right)^2 + \left(\dot{y}_h + c_y y_e + y_{VF}\right)^2} \\
\omega_m &= \dot{\alpha}_{VI} + c_\theta \theta_e
\end{aligned}
\] (3.16)

Here, \(x_{VF} = \text{sign}(x_e) c_{VF} \sqrt{x_e^2 + y_e^2}\)

\(y_{VF} = \text{sign}(y_e) c_{VF} \sqrt{x_e^2 + y_e^2}\)

\(\alpha_{VI} = \arctan\left(\frac{\dot{y}_h + c_y y_e + \text{sign}(y_e) y_{e} c_{VF} \sqrt{x_e^2 + y_e^2}}{\dot{x}_h + c_x x_e + \text{sign}(x_e) x_e c_{VF} \sqrt{x_e^2 + y_e^2}}\right)\)

In controller (3.16), the actual inputs are the position errors \(x_e, y_e\) in the cartesian coordinate, and the rotation error \(\theta_e\). The position error signal is the difference between the current position and the initial position of the end of the Phantom stylus, which can be obtained from the encoders on the phantom. The rotation error is the difference between the current posture and the desired posture of the mobile robot, which can be read from the gyroscope on the mobile robot.

### 3.3.2 Controller based on Polar Coordinate

The polar coordinate arrangement of the mobile assistive robot is shown in Fig. 3.4:
Consider the following kinematics model for the nonholonomic mobile robot:

$$
\dot{P}_m = \begin{bmatrix}
    \dot{x}_m \\
    \dot{y}_m \\
    \dot{\theta}_m
\end{bmatrix} = \begin{bmatrix}
    \cos \theta_m & 0 \\
    \sin \theta_m & 0 \\
    0 & 1
\end{bmatrix} \begin{bmatrix}
    v_m \\
    \omega_m
\end{bmatrix},
$$  \hspace{1cm} (3.17)

where $\theta_m$ is the angle between the position in which the robot is heading and the positive $X$-axis. $v_m$ is the translational velocity and $\omega_m$ is the rotational velocity of the robot.
We define the errors as:

\[
\begin{bmatrix}
    r_e \\
    \theta_e \\
    \phi_e \\
    \varphi_e
\end{bmatrix} =
\begin{bmatrix}
    \sqrt{(x - x_m)^2 + (y_h - y_m)^2} \\
    \theta_r - \theta_m \\
    \theta_{re} - \theta_m \\
    \theta_{re} - \theta_r
\end{bmatrix},
\] (3.18)

Here \( r_e \) is the distance between the robot’s current position and the human’s position. \( \theta_e \) is the angle between the human input’s direction (\( \theta_r \)) and the mobile robot’s orientation(\( \theta_m \)). \( \theta_{re} \) is the angle between \( r_e \) and the positive X-axis. \( \phi_e \) is the error between the angle of \( r_e \) (\( \theta_{re} \)) and the mobile robot’s orientation(\( \theta_m \)). \( \varphi_e \) is the error between the direction of \( r_e \) (\( \theta_{re} \)) and the human input’s direction (\( \theta_r \)).

If we assume the human’s velocity and the rotational velocity is \( v_r \) and \( \omega_r \), the derivative of the errors \( \dot{\phi}_e \) and \( \dot{\theta}_e \) can be written as

\[
\dot{\phi}_e = \dot{\theta}_{re} - \dot{\theta}_m = v_m \frac{\sin \phi_e}{r_e} - v_r \frac{\sin \varphi_e}{r_e} - \omega_m,
\] (3.19)

\[
\dot{\theta}_e = \dot{\theta}_r - \dot{\theta}_m = \omega_r - \omega_m.
\] (3.20)

Considering the following candidate Lyaponuv function:

\[
V = V_1 + V_2 = \frac{1}{2} r_e^2 + \frac{1}{2} \phi_e^2 + \frac{1}{2} \theta_e^2.
\] (3.21)

First, the derivative of \( V_1 = \frac{1}{2} r_e^2 \) is:

\[
\dot{V}_1 = r_e \dot{r}_e = r_e (v_r \cos \varphi_e - v_m \cos \phi_e).
\] (3.22)
CHAPTER 3. MOBILE ROBOT CONTROLLER DESIGN

If we make

\[ v_m = \frac{v_r \cos \varphi_e}{\cos \phi_e} + \lambda_v r_e \cos \phi_e. \tag{3.23} \]

Then \( \dot{V}_1 = -\lambda_v r_e^2 \cos^2 \phi_e < 0 \), where \( \lambda_v (\lambda_v > 0) \) is the gain for \( v_m \) which can control the amplitude of \( v_m \). For the derivative of \( V_2 = \frac{1}{2} \phi_e^2 + \frac{1}{2} \theta_e^2 \):

\[ \dot{V}_2 = \phi_e (\omega_{re} - \omega_m) + \theta_e (\omega_r - \omega_m). \tag{3.24} \]

Here, \( \omega_{re} \) is the rotational velocity of \( r_e \). \( \omega_{re} = \dot{\theta}_{re} \).

If we make \( \omega_m \) as

\[ \omega_m = \frac{\omega_{re} \phi_e + \theta_e \omega_r + \lambda_\omega \phi_e^2}{\theta_e + \phi_e}, \tag{3.25} \]

where \( \omega_{re} \) and \( \omega_r \) are known. \( \lambda_\omega \) is the gain for \( v_m \).

We have \( \dot{V}_2 = -\lambda_\omega \phi_e^2 < 0 \), where \( \lambda_\omega (\lambda_\omega > 0) \) is the gain for \( \omega_m \) which can control the amplitude of \( \omega_m \).

By combining (3.23) and (3.25), we make the system stable (\( \dot{V} < 0 \)) and have the controller for the mobile robots as:

\[
\begin{align*}
\dot{v}_m &= \frac{v_r \cos \varphi_e}{\cos \phi_e} + \lambda_v r_e \cos \phi_e \\
\dot{\omega}_m &= \omega_{re} \phi_e + \theta_e \omega_r + \lambda_\omega \phi_e^2 \\
&\quad \quad + \lambda_\omega \phi_e^2 \\
&\quad \quad \quad \div (\theta_e + \phi_e).
\end{align*}
\tag{3.26}
\]
3.4 Adaptive Windowing Velocity Estimation Based on Fuzzy Logic

One important issue for the designed controller is the estimation of the Phantom Omni’s stylus velocity in (3.16) and (3.26). The Phantom Omni only provides us the encoder information that can be converted to the position in the Cartesian space. The velocity information is derived from this position information, which is so called the velocity estimation problem. This problem exists in a very wide range, especially for the control problems, such as motor control, robot manipulator control, etc [71].

There are two general approaches to solve the velocity estimation problem: model-based or non-model-based techniques. The detailed literature review can be found in [72].

How to choose a suitable method to solve the velocity estimation problem depends on the specific system. For the proposed system, several perspectives are considered as distinguished features when compared to the other systems.

- The refresh rate of the mobile robot control loop is 100Hz - a relatively low refresh rate compared to the haptic device.

- The developing software is Visual Studio C++ 2005, which is a powerful platform for implementing the non-model-based techniques, while the MATLAB/Simulink Environment is more suitable for the implementation of the model-based approaches.

Based on the above considerations, a non-model-based method is chosen. The velocity $\hat{V}$ can be calculated using the first order differentiation as

$$\hat{V} = \frac{P_k - P_{k-n}}{nT}$$  \hspace{1cm} (3.27)
Here, $P_k$ is the current position of the mobile robot. $P_{k-n}$ is the next $n$th position of the mobile robot. $n$ is the size of the window to estimate the velocity, and $T$ is the sampling time.

One of the non-model-based methods for the haptic device is the discrete adaptive window technique [73]. The main idea of this technique is to use an adaptive law based on the uncertainty band, to change the size of the window for the velocity estimation. However, this adaptive law cannot be implemented on our proposed system, because since the refresh rate of the mobile control loop is low, the phantom velocity will change greatly during the sampling period compared to the high refresh rate system. It is very hard to set a reasonable uncertainty band for this type of system.

In this thesis, a novel adaptive law for changing the size of the estimation window is proposed. The current position error and its first order differentiation value are merged together by the fuzzy logic to adapt the size of the estimation window. Due to certain distinguishing characteristics of fuzzy logic, such as approximation (rather than accuracy), multiple inputs, etc., this method is especially suitable for merging these two kinds of data together [74–76]. This algorithm is the combination of the fuzzy logic velocity estimation method [77] and the adaptive window method [73]. Fig.3.5 shows the detail diagram of the proposed method.

![Diagram of fuzzy logic windowing velocity estimation method](image)

**Figure 3.5:** Diagram of fuzzy logic windowing velocity estimation method
3.4.1 Fuzzification

The first step in the fuzzy logic algorithm is the fuzzification. The two inputs are the position error between the current mobile robot and the human input, and its first order differentiation value. The outputs are the classified values that are obtained from these two inputs, belonging to the specific membership functions. A simple fuzzification membership function has been chosen for the two inputs, as shown in Fig.3.6. The simple triangle function is chosen as the membership function.

In Fig.3.6, NB, NM, ZE, PM and PB are the abbreviations for negative big, negative medium, zero, positive medium and positive big respectively.

3.4.2 Fuzzy Rules

The fuzzy rules are derived from the real time operation in our proposed system: The main factors that affect the window size in the velocity estimation problem are the position error \( P_e \) and its first order differentiation value (the change of position error) \( \dot{P}_e \). The windows size is changed by the different increasing and decreasing trend between these two values. It is worth mentioning that the fuzzy output and
the window size have a reverse relationship. That means that when the fuzzy output is PB, the window size is small and when fuzzy output is NB, the window size is big. The rules are listed below:

1. When \( P_e \) and \( \dot{P}_e \) are PB, the distance between the human hand and the mobile robot is large and will increase greatly due to the increasing \( \dot{P}_e \). The window size should therefore be small enough to catch the rapid change in the velocity, which corresponds to the PB of the fuzzy output.

2. When \( P_e \) and \( \dot{P}_e \) are NB, the distance between the human hand and the mobile robot is nearly constant and will stay the same in the near future. The window size should therefore be large enough to cancel the noise while calculating the velocity.

3. When \( P_e \) is PB, but \( \dot{P}_e \) is NB, the distance between the human hand and the mobile robot is large but will stay the same in the near future. The window size should therefore be medium to both cancel the noise and get the rapid change of the velocity and vice versa.

4. When \( P_e \) and \( \dot{P}_e \) are ZE, the distance is in the middle and the change trend of the distance is medium. The window size should therefore also be medium to cancel the noise and get the change in velocity.

The above rules do not cover all the possible situations. The rest of the scenario can be derived from the above rules. A summary of the fuzzy logic rules is shown in Table 3.1.
3.4.3 Defuzzification

A simple weighted average defuzzification method has been adopted to calculate the output (window size for the velocity estimation). The equation is:

$$n_{\text{window}} = n_{\text{max}} - k_{\text{win}} \left\lfloor \frac{\sum f_m(\bar{\phi})\bar{\phi}}{\sum f_m(\bar{\phi})} \right\rfloor$$  \hspace{1cm} (3.28)

Here $n_{\text{window}}$ is the size of the velocity estimation window which is calculated by the integer value of the fuzzy output value. $n_{\text{max}}$ is the max window size which is chosen by trial and error. $k_{\text{win}}$ is the output gain for the fuzzy logic, and $f_m$ is the fuzzy membership function. $\bar{\phi}$ is the average value of the membership function.

3.4.4 Velocity Calculation

The size of the velocity estimation window has been given by the fuzzy logic algorithm. The estimation of the velocity can then be calculated by the equation:

$$\dot{V} = \frac{P_k - P_{k-n_{\text{window}}}}{n_{\text{window}}T}$$  \hspace{1cm} (3.29)
3.5 The Safety Control for the Manipulation of the Assistive Robot

As mentioned in Chapter 1, the safety consideration of the assistive robot is crucial when designing the control algorithm. After some initial experiments, we found that the mobile robot’s rotational velocity usually plays an important role in the movement of the robot. This rotational velocity highly depends on the base angle of the Phantom. The larger the base angle is, the faster the mobile robot turns. However, for the Phantom Omni device, the base angle is constrained to around $\pm 60^\circ$ by the mechanical property (shown in Fig.3.7). Consequently, it is impossible for the mobile robot to turn faster when the base angle of the Phantom comes to $\pm 60^\circ$. In addition, if the base angle of the Phantom is around $\pm 60^\circ$, a collision between the Phantom base and Phantom arm may occur. This collision should be avoided for the safe operation of the assistive robot. Therefore, in designing the rotational velocity controller of the mobile robot, we need to consider this issue and design the corresponding control algorithm.

Based on the analysis of the problem, a revised constraints method is adopted to solve this problem. The constraints method is usually applied in the high number of degrees of freedom robot manipulator to find a collision-free trajectory [78] [79].

The first part of our method uses a similar approach. The optimization problem of finding the rotational velocity $\omega_m$ can be stated as follows:

Minimize $f_r = \frac{1}{2} \|\omega_m - \omega_{goal}\|$,  

Subject to $\omega_{min} < \omega < \omega_{max}$.
Here $\omega_{\text{goal}}$ is the desired rotational velocity control input, and $\omega_{\text{max}}$ and $\omega_{\text{min}}$ are the maximum and minimum rotational velocity of the mobile robot respectively. The following theorem gives the condition for the $\omega_{\text{goal}}$.

**Theorem 1** If the desired rotational velocity control input is:

$$\omega_{\text{goal}} > \omega_{\text{re}} - \varepsilon \frac{\phi_e^s - \phi_e}{\phi_e^s - \phi_e^i},$$

where $\phi_e^s$ is the security angle that must be respected (shown in Fig.(3.7)). $\phi_e^i$ is the angle from which the constraint becomes active (shown in Fig.(3.7)). $\varepsilon$ is the coefficient that adapts the convergence speed. The Phantom cannot reach its maximum base angle.

**Proof:** The goal that Phantom cannot reach its maximum angle with $t \to \infty$ can be expressed as:

$$\phi_e < \phi_e^s.$$
The equation can be made stricter in order to maintain safety:

\[ \dot{\phi}_e < \phi^* - F(0)e^{-\frac{\varepsilon t}{\phi^*_e - \phi^*_e}}, \]

where \( F(t), t \in R \) is a non-negative and monotone increasing function. After a manipulation of the terms, we have:

\[ (\phi^*_e - \phi_e)e^{\frac{\varepsilon t}{\phi^*_e - \phi^*_e}} > F(0). \]

If we make

\[ F(t) = (\phi^*_e - \phi_e)e^{\frac{\varepsilon t}{\phi^*_e - \phi^*_e}}. \]

Because \( F(t) \) is non-negative and monotone increasing, we have:

\[ \dot{F}(t) = [-\dot{\phi}_e + (\phi^*_e - \phi_e)\varepsilon]\frac{e^{-\frac{\varepsilon t}{\phi^*_e - \phi^*_e}}}{\phi^*_e - \phi^*_e} > 0. \]

We get:

\[ \dot{\phi}_e < (\phi^*_e - \phi_e)\varepsilon. \]

From (3.19), we have \( \dot{\phi}_e = \omega_{re} - \omega_m \). That is

\[ \omega_{re} - \omega_m < \varepsilon\frac{\phi^*_e - \phi_e}{\phi^*_e - \phi^*_e}. \]

So we finally get the \( \omega_{goal} \) as:

\[ \omega_{goal} > \omega_{re} - \varepsilon\frac{\phi^*_e - \phi_e}{\phi^*_e - \phi^*_e}. \]

By combining (3.30) with the constraint of the maximum rotational velocity of the mobile robot, we get the actual control input \( \omega_m \).
3.6 Haptic Rendering Algorithm

In the previous sections, we have developed the controllers for the mobile robot, but there is still no haptic feedback for the user. It is hard for the human user to know the positional error between the human and the robot, or whether he/she should wait for the response of the mobile robot. Therefore, the haptic rendering algorithm based on the error information should be added to the system. In the proposed system, the goal of the haptic rendering algorithm is to compute the feedback force based on the measurement of the operators motions. In current literature, the sensing, collision detection, surface point determination, force calculation, etc. has been extensively investigated. For our application, there is no virtual object, therefore only the force calculation is considered. For this reason, we develop a sample haptic rendering algorithm based on Hooke’s Law, which utilizes the position error information for the system. This haptic algorithm acts as an indicator that an increasing error simultaneously increases the force feedback. The human user feels the force feedback and waits for the response of the mobile robot to follow the human movement. This process is the same as humans interacting with each other. Fig.3.8 shows the decomposition of the feedback force.

The absolute value of the feedback force along the X axis \( f_x \) and the Y axis \( f_y \) can be calculated as

\[
|f| = k_f \sqrt{(x_r - x_m)^2 + (y_r - y_m)^2},
\]

where, \( k_f \) is the feedback force gain.

If \( \phi_e \) goes into the unsafe zone in the Phantom Omni, the feedback force gain \( k_f \) will change to \( k_f^s = \frac{\phi_e^s - \phi_e^i}{\phi_e^s - \phi_e^i} \). This means when \( \phi_e \rightarrow \phi_e^s \) the feedback force will be larger.
Therefore, the absolute value ($|f|$) and the direction ($\theta_f$) of the feedback force are

$$|f| = \begin{cases} k_f \sqrt{(x_r - x_m)^2 + (y_r - y_m)^2} & \phi_e > \phi_e^i \\ k_s^f \sqrt{(x_r - x_m)^2 + (y_r - y_m)^2} & \phi_e^* > \phi_e > \phi_e^i \end{cases}$$

$$\theta_f = \arccos \frac{x_r - x_m}{\sqrt{(x_r - x_m)^2 + (y_r - y_m)^2}}$$

3.7 Summary and Conclusion

This chapter mainly introduces two different tracking controllers for the mobile robot. Two controllers are designed in rectangular coordinate and polar coordinate systems, respectively. Besides the design of the controllers, several technical problems are presented and solved. A velocity estimation problem is solved by the adaptive windowing
method, based on fuzzy logic. A safety control problem is solved by a revised constraints method. In addition, a haptic rendering algorithm based on Hook’s Law is developed to provide the haptic feedback for the user.
Chapter 4

The DVZ Obstacle Avoidance Algorithm

4.1 Introduction

In Chapter 3, two kinds of tracking controllers for the mobile robot were developed. These tracking controllers are part of the shared control diagram. The other important part is the autonomous obstacle avoidance algorithm. Because the mobile assistive robot is employed to assist users in a clustered environment, a stable obstacle avoidance algorithm is essential to ensure the safe operation of the robot.

In the current literature, many approaches have been applied to solve the obstacle avoidance problem for mobile robots [80]. From a sensing point of view, there are two main types of approaches: the algorithms based on a prior map and the algorithms based on real time sensing. For the first type, a map is used to construct the potential field that can generate a virtual force to drive the mobile robot to avoid the obstacles. This type of approach is mainly used for planning robot motions in a large area where the objects positions are fixed [81–84]. For the second type, range sensors are used to sense the obstacles around the robot. The driving commands allowing the robot to move away from the obstacles are then generated from the sensing information. This type of approach is used for planning robot motions in small areas where object
positions are usually unknown [85–89].

When comparing these two types of approaches, we find that the second group is more suitable for our obstacle avoidance algorithm. In our application, the environment varies a great deal, and it is very hard to get the prior map. Also, the mobile assistive robot is commanded by the user and does not initially know the final goal. Accurate real-time sensing capabilities are a necessity at this point. Since the users of the robotic system are senior and disabled people, a smooth maneuverability is also necessary for the obstacle avoidance algorithm. In addition, this obstacle avoidance algorithm should be proved stable by the Lyapunov stability theory, since we need this feature to analyze the stability of the shared controller (Chapter 5).

Based on the requirements and analysis, a deformable virtual zone (DVZ) principle is used to design the obstacle avoidance algorithm.

### 4.2 The Deformable Virtual Zone (DVZ) Principle

The deformable virtual zone (DVZ) principle is first proposed by Zapata [89–91] to design the obstacle avoidance algorithm. The main idea of this principle is to introduce a virtual zone around a robot, which helps the robot interact with its surroundings. The robot is commanded by the deformation of this virtual zone, which allows it to avoid potential risks. Compared to other static techniques, this DVZ is dynamic. The shape and geometry of the zone depends on the mobile robots translational and rotational velocity.

#### 4.2.1 General Statement of the DVZ Principle

The different DVZ definitions can be found in Fig. 4.1.
The controlled DVZ is defined as $\Xi_h$, the shape of which is controlled by $U$, where $U$ is the velocity of the robot $U = [v_m, \omega_m]$. We have the following function:

$$\Xi_h = \rho(U).$$  \hspace{1cm} (4.1)

When the mobile robot is moving, sensors on the mobile robot sense its surroundings. If there are no obstacles, the sensing boundary is defined as $\Theta$. If there are obstacles, the sensing boundary is defined as $\Psi$. The difference between these two boundaries can be defined as $I$, which is the deformation of the sensing boundary. Here we have:

$$I = \Psi - \Theta.$$  \hspace{1cm} (4.2)

We define $\Xi_I$ as a DVZ depending on sensor boundary deformation $I$ and control
CHAPTER 4. THE DVZ OBSTACLE AVOIDANCE ALGORITHM

by $U$.

\[ \Xi_I = \beta(I, U). \]  
(4.3)

The deformation of DVZ can be written as:

\[ \Delta = \Xi_h - \Xi_I = \rho(U) - \beta(I, U). \]  
(4.4)

The deformation function $\Delta$ is the difference between the sensor boundary deformation $I$ and the controlled DVZ.

From (4.4), we know that if we want to make the robot avoid obstacles, we need to decrease $\Delta$. When $\Delta$ decreases, $\Xi_I$ and $\Xi_h$ will get close to each other and there will be less obstacles within the DVZ around the robot. Therefore, if we want to use (4.4) to design the obstacle avoidance controller, we have to differentiate $\Delta$ along its grads direction to control $v_m$ and $\omega_m$ for the mobile robot. With this controller, $\Delta$ can be minimized to avoid obstacles.

To calculate the control signal needed for the mobile robot to avoid obstacles, differentiate $\Delta$ with respect to $U$:

\[ \dot{\Delta} = -\nabla_U [\rho] \dot{U} + \nabla_U [\beta] \dot{U} + \nabla_I [\beta] \dot{I}. \]  
(4.5)

Here, $\nabla_U$ is the differentiation operator with respect to $U$. $\nabla_I$ is the differentiation operator with respect to $I$. (4.5) is a functional equation expression and has to be parameterized to be differentiable. This parameterization process is to find a specific mathematical expression for the controlled DVZ.
4.3 The DVZ for the Mobile Assistive Robot

4.3.1 The parameterized controlled DVZ

The nonholonomic mobile robot has as a constraint that the robots translational velocity always needs to be orthogonal to the driven wheel axis. Because of this constraint, an elliptical shape (shown in Fig. 4.2) is chosen as the shape of the controlled DVZ. This shape focuses mainly on the front, and partly on both sides of the robot. When the robot turns, the direction of the DVZ changes according to the robots rotational velocity. This change is suitable and practical for the robot to sense its environment, just as when we turn a vehicle, the focus is in the direction of the turning side rather than straight ahead.

We assume \( D = [x_D, y_D]^T \) is a point on the ellipse that has axes \( a_x \) and \( b_y \). Unlike in the current literature about DVZ, a polar coordinate is applied to formulate the mathematical expression of the DVZ. By using the polar coordinate, the calculation
of the derivative can be largely simplified. The equation describing the shape of the controlled DVZ can be expressed as:

\[
\begin{align*}
    x_D &= a_x \cos \alpha_D \cos \varphi - b_y \sin \alpha_D \sin \varphi \\
    y_D &= a_x \cos \alpha_D \sin \varphi + b_y \sin \alpha_D \cos \varphi.
\end{align*}
\]  

Here, \(a_x\) and \(b_y\) are the semi-minor and semi-major axes of the ellipse, respectively. They are directly proportional to the robot’s translational velocity. We have \(a_x = k_{a_x} v_m\) and \(b_y = k_{b_y} v_m\). \(k_{a_x}\) and \(k_{b_y}\) \((k_{a_x}, k_{b_y} > 0)\) are the proportional ratios. These two equations mean that when the robot moves faster, the ellipse becomes larger.

\(\alpha_D\) is the angle between the robot’s heading direction and the line that connects the point on the ellipse and the origin, and \(\varphi\) is the angle between the \(b_y\) axis and the robot’s heading direction. \(\varphi\) is directly proportional to the rotational velocity. We have \(\varphi = k_{\omega_m} \omega_m\), \(k_{\omega_m} > 0\) as the proportional ratio. The origin of this ellipse is the mobile robot.

The distance between the robot and the point on the DVZ can be defined as \(d_h\). Because the DVZ equations are defined in polar coordinates, we can easily calculate \(d_h\) as:

\[
d_h(\alpha_D) = \sqrt{x_D^2 + y_D^2} = \sqrt{a_x^2 \cos^2 \alpha_D + b_y^2 \sin^2 \alpha_D}.
\]  

4.3.2 The Deformation of the Controlled DVZ

With the mathematical expression of the controlled DVZ, we can precisely describe the DVZ deformation. An intrusion ratio is introduced in (6.1) to record the deformation of DVZ around the mobile robot:
\[ I_D = \int_{-\pi}^{\pi} \frac{d_h(\alpha_D) - d(\alpha_D)}{d_h(\alpha_D)} d\alpha_D. \] (4.8)

Here, \(d(\alpha_D)\) is the distance between the robot and the obstacle that is recognized by the sonar sensor on the robot. Also, the sampling rate of the sonar sensor is the same as the refresh rate of the robot, which is 100 Hz. This refresh rate is relatively low, ensuring that the noise does not affect the sampling results. The intrusion ratio is introduced to quantify the DVZ deformation. We can design the controller of the mobile robot to minimize \(I_D\) to ensure the robot can avoid the obstacles.

In addition, we introduce an average angle \(\overline{\theta}_D\) to indicate the current orientation of the obstacles relative to the mobile robot.

\[ \overline{\theta}_D = \frac{\int_{-\pi}^{\pi} (d_h(\alpha_D) - d(\alpha_D))\alpha_D d\alpha_D}{\int_{-\pi}^{\pi} (d_h(\alpha_D) - d(\alpha_D))d\alpha_D} + \theta_m. \] (4.9)

From (4.9), we can see that the average angle \(\overline{\theta}_D\) is the weighted average of \(\alpha_D\). This value gives the direction in which most of the obstacles are located. Using the controller of the mobile robot, we can command the robot to move away from this direction.

### 4.3.3 The Obstacle Avoidance Controller

The following Lyapunov function candidate is chosen to design the controller:

\[ V_D = V_{D,I} + V_{D,\theta} \]
\[ = \frac{1}{2} I_D^2 + \frac{1}{2} (\theta_m - (-\overline{\theta}_D))^2. \] (4.10)

The derivation of \(V_D\) is:
\[ \dot{V}_D = \dot{V}_{D,I} + \dot{V}_{D,\phi} \]
\[ = I_D \dot{I}_D + (\theta_m + \vec{\theta}_D)(\dot{\theta}_m + \dot{\vec{\theta}}_D) \]  \hspace{1cm} (4.11)

and \( \dot{V}_{D,I} \) can be written as:

\[ \dot{V}_{D,I} = I_D \dot{I}_D \]
\[ = I_D (J_{1D}^{x_m} v_{x_m} + J_{1D}^{y_m} v_{y_m} + J_{1D}^{pob} v_{ob}). \]  \hspace{1cm} (4.12)

Here, \( J_{1D}^{x_m} \) and \( J_{1D}^{y_m} \) is the Jacobian function of the X-axis and Y-axis velocities of the intrusion ratio’s time derivation. We note \( v_{x_m} = \dot{x}_m, v_{y_m} = \dot{y}_m \). \( J_{1D}^{pob} \) is the Jacobian function for the intrusion ratios time derivation on the obstacles velocity. \( v_{ob} \) is the obstacles velocity. Because the sample time of the sonar sensor is very short compared to the movement of the obstacles, we can assume that the obstacles are not in movement when the sonar sensor detects them. Therefore, the obstacles velocity \( v_{ob} = 0 \).

From (4.11) and (4.12), we can choose the translational velocity controller in each axis as:

\[ v_{x_m} = -k_{v_{x_m}} J_{1D}^{x_m}, \]  \hspace{1cm} (4.13)

and

\[ v_{y_m} = -k_{v_{y_m}} J_{1D}^{y_m}. \]  \hspace{1cm} (4.14)

Here, \( k_{v_{x_m}} \) and \( k_{v_{y_m}} \) are the gains for \( v_{x_m} \) and \( v_{y_m} \), respectively. These two gains
can control the amplitudes of $v_{xm}$ and $v_{ym}$, respectively. The Jacobian functions $J^x_{ID}$ and $J^y_{ID}$ can be expressed as:

\[
\begin{align*}
J^x_{ID} &= \frac{\partial I_D}{\partial x_m} = \int_{-\pi}^{\pi} -\frac{x_m - x_{ob}(\alpha_D)}{d_i(\alpha_D)d(\alpha_D)} d\alpha_D, \\
J^y_{ID} &= \frac{\partial I_D}{\partial y_m} = \int_{-\pi}^{\pi} -\frac{y_m - y_{ob}(\alpha_D)}{d_i(\alpha_D)d(\alpha_D)} d\alpha_D.
\end{align*}
\]

(4.15)

In (4.15), $[x_m, y_m]$ is the position of the mobile robot. $[x_{ob}(\alpha_D), y_{ob}(\alpha_D)]$ is the position of the obstacle at an angle of $\alpha_D$ to the robot.

If we combine the translational velocity controller in each axis, we have the translational velocity controller $v_o$ for the obstacle avoidance controller:

\[
v_o = v_{xm} \cos \theta_m + v_{ym} \sin \theta_m.
\]

(4.16)

The rotational velocity controller $\omega_o$ for the obstacle avoidance controller is:

\[
\omega_o = -k_{\omega_o}(\theta_m + \bar{\theta}_D) - \dot{\bar{\theta}}_D.
\]

(4.17)

Here, $k_{\omega_o}$ is the gain that can control the amplitude of $\omega_o$.

### 4.4 Summary and Conclusion

In this chapter, a deformable virtual zone (DVZ) principle is adopted to design the obstacle avoidance algorithm. This DVZ principle based obstacle avoidance algorithm is a Lyapunov stable obstacle avoidance algorithm, which is suitable for the shared controllers stable analysis. An improved calculation method for the DVZ, based on the polar coordinate is introduced to simplify the DVZ calculation.
Chapter 5

Adaptive Servo Level Shared Control
Design

5.1 Introduction

In Chapters 3 and 4, two tracking controllers and an obstacle avoidance algorithm are introduced separately. In this chapter, we will introduce the design and stability analysis of an adaptive servo level shared control for the mobile assistive robot by using these controllers and algorithms.

The goal of the shared controller is to dynamically assign the control weight between the human and the autonomous control in order to find a stable shared control output, based on a real-time situation. For the proposed mobile assistive robot, the target users are senior and disabled people who have limited mobility and cognitive levels. Due to these limited mobility and cognitive levels, or even because of irrational behavior, the stable and safe operation of the system under human control cannot be guaranteed. For example, senior people with reduced visual capacity may not notice the obstacles around when operating the assistive robot. Disabled people with reduced mobility may not react fast enough to command the robot in order to avoid
the obstacles. All of these situations need to be considered in the design of a stable and safe controller for the mobile assistive robot.

The stability analysis of the proposed shared controller is another important aspect of this chapter. As mentioned in Chapter 1, the proposed shared controller should be well analyzed in terms of stability. From a control theory point of view, before being applied, any controller should be analyzed for its stability. With this analysis, the stable range of the controller should be found.

The organization of this chapter is as follows: First, we give a general view of the adaptive servo level shared control. Then we analyze the stability of the shared controller in different situations. At last, based on this stability analysis, a simple shared control algorithm is proposed to ensure the stable operation of the mobile assistive robot.

5.2 Overview of the Adaptive Shared Control Framework

The proposed adaptive shared control framework is shown in Fig. 5.1. The human control block sends the control signal with the allocation weight $1 - \alpha_s$. This control signal is given by the tracking controller in Chapter 3. The autonomous control block sends the control signal with the allocation weight $\alpha_s$. This control signal is calculated by the obstacle avoidance algorithm in Chapter 4. Both the human control and the autonomous control signals are collected by the shared control block. The shared control block outputs the shared control signal to the mobile robot. An adaption law based on the human and autonomous control inputs is used to change the allocation weights for each control input in the shared control block.
5.3 Stability Analysis of the Adaptive Servo Level Shared Control

This section addresses the adaptive servo-level shared control stability analysis for the proposed mobile assistive robot. The objective of this shared controller is to dynamically assign the allocation weight to the human and to the autonomous control, according to the real-time situation. Since the change of this allocation weight affects the shared control output, this change needs to make the system stable. For this reason, a stability analysis should be carried out to find the stable range of the system. Since our mobile assistive robot is a nonlinear system, and the user control input is unpredictable, it is very hard to use a conventional way of analyzing the stability of the system.

There are two interesting features for our assistive robot system:

1. The shared controller (1.1) is the linear combination of human and robot control inputs. If we draw a straight line between $u_r$ and $u_h$, $u_s$ is just on this line.

2. The stable ranges of tracking controller and obstacle avoidance controller are convex. Since the stable range of $v$ and $\omega$ of these controllers are continuous,
the total stable range of these two controllers is convex.

Inspired by these two features, we find that a convex analysis and set theory can help us analyze the shared controller in our system. These two tools are suitable to describe characteristics of a nonlinear system. Set theory is a good tool to use to discuss the relative positions of the two controller's outputs: the position of the unknown obstacles and the unpredictable human behavior.

Before applying convexity to analyze the stability of the shared controller, we introduce the basic concept of a convex analysis. First, we introduce the convexity in general, then we introduce the useful concepts related to convexity in this thesis.

5.3.1 Introduction to Convex Analysis

In the field of science, convexity is a basic notion in geometry. It describes the basic characteristics of a graph. A convex graph has a curving out or bulging outward feature. The antonym of convexity is concavity. A concave graph has a curving in or hollowed inward feature. A simple illustration in Fig. 5.2 shows the difference between convexity and concavity. The two shapes in the first row are the convex shapes. The two shapes in the second row are the concave shapes.

Convexity does not only exist in geometry, as it is widely used in other areas of mathematics: functional analysis, complex analysis, calculus of variations, graph theory, partial differential equations, discrete mathematics, algebraic geometry, probability theory, coding theory, and many other fields. Thousands of years ago, convexity was already considered by Greek philosophers, Egyptians and Babylonians [92].

In our application, the convex concept describes the property of the control input sets. These sets contain the stable control inputs of the nonlinear system. For our system, we have two kinds of control inputs: 1. Tracking control input 2. Obstacle
avoidance control input. The convex concept will be used as an important feature to discuss the relationships between the control input sets. These relationships will significantly influence the stability of the shared controller. Before we analyze the stability of the shared controller, some useful convex concepts are introduced below.

**Convex set**

If a line segment between any two points in a nonempty set $A$ lies in $A$, this nonempty set $A$ is convex. If we express in the formal mathematical expression, the definition is: For any $x_1$ and $x_2 \in A$, and any $\theta \in [0, 1], \theta \in R$, we have

$$\theta x_1 + (1 - \theta)x_2 \in A \quad (5.1)$$

The set $A$ is a convex set.

**Convex combination**

For $x_1, x_2, \cdots, x_k$ and $\theta_1, \theta_2, \ldots, \theta_k$ any $x$ has a form of:
\[ x = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_k x_k. \] (5.2)

\( x \) is called a convex combination of \( x_1, x_2, \ldots, x_k \).

**Operations preserving convexity**

In our application of convexity, we need to discuss the relationships of different sets. Therefore, we need to know some operations that preserve convexity. In general, there are four operations that allow this:

- intersections
- affine functions
- perspective functions
- linear-fractional functions

In these four operations, we primarily use the first operation: intersection. The idea is that any convex set that intersects with others will form a new intersection convex set.

**5.3.2 Stability Analysis of the Proposed Shared Controller**

The shared controller can be formulated as:

\[ u_s = (1 - \alpha_s)u_h + \alpha_s u_r. \] (5.3)

Here, \( u_s = [v_s, w_s] \) is the shared control signal for the mobile robot (\( v_s \) and \( w_s \) are the translational and rotational velocities). \( u_h = [v_m, w_m] \) and \( u_r = [v_o, w_o] \) are the
human and autonomous control inputs, respectively, and $\alpha_s$ is the allocation weight with domain $0 < \alpha_s < 1$. We know that $u_s$ is a convex combination of $u_h$ and $u_r$.

A generalized expression for the feasible human input set $U_h$ can be defined as $U_h = \{u_h : \dot{V}_h(u_h) < 0\}$. For the human input $u_h$, there is one possible solution. Therefore, we have $u_h \in U_h$. As mentioned at the beginning of this section, in a real situation where the stable range of the tracking controller is continuous, we can assume that the set $U_h$ is convex. The feasible autonomous control input set $U_r$ can similarly be defined as $U_r = \{u_r : \dot{V}_r(u_r) < 0\}$. The obstacle avoidance controller $u_r$ is one possible solution, where we have $u_r \in U_r$. In a real situation, we can also assume the set $U_r$ is convex. We can see that $u_s$ is a point on the closed line segment passing through $u_h$ and $u_r$. The set for points on this closed line segment can be defined as $U_{sl} = \{u_s : u_s = (1 - \alpha_s)u_h + \alpha_s u_r | u_s \in U_s, u_h \in U_h, 0 < \alpha_s < 1\}$.

There are four relationships between $U_h$ and $U_r$ that apply to real situations of the developed assistive robot.

1. $U_h \subset U_r$. This is a stable situation. The feasible set of human inputs is a subset of the feasible set of autonomous control input. The human input not only makes the tracking error smaller, but also avoids obstacles. If the user acts perfectly and makes no mistakes, we simply put $\alpha_s = 0$ to let the user control the system. Fig. 5.3 shows this situation.

2. $U_r \subset U_h$. The feasible set of autonomous control inputs is a subset of the feasible set of human inputs. Some of the human inputs can achieve both goals, but some inputs cannot. In a real application, this situation means there are lots of obstacles around the robot. It is very hard for the user to find a feasible control input to achieve both goals. Therefore, only by assigning an allocation weight $\alpha_s$ is the shared control signal able to compensate for user’s deficiency.
and achieve both goals. From a control theory point of view, this is to find a range of $\alpha_s$ to stabilize the system. The feasible set of $u_s$ is $U_s = U_r \cap U_{sl}$ which is the line segment in $U_r$. The allocation weight $\alpha_s$ that can stabilize the shared controller should be $\alpha_{sl} < \alpha_s < 1$. Here, $\alpha_{sl}$ is the lower bound of $\alpha_s$. Any $\alpha_s$ within this range is a feasible solution for the shared controller. Otherwise, the shared controller cannot achieve both goals. Fig. 5.4 shows this situation.

3. $U_r \cap U_h \neq \emptyset$, $U_r \not\subset U_h$ and $U_h \not\subset U_r$. In this situation, the feasible set of human inputs and the feasible set of autonomous control inputs intersect. In a real application, this situation is the same as the second situation, but more complicated. The feasible set of $u_s$ is $U_s = U_r \cap U_h \cap U_{sl}$, which is the line segment in $U_r \cap U_h$. The stable allocation weight $\alpha_s$ should be $\alpha_{sl} < \alpha_s < \alpha_{su}$. Here, $\alpha_{su}$ is the upper bound of $\alpha_s$. Fig. 5.5 shows this situation.
**Figure 5.4:** Diagram of the stability analysis for shared control: $U_r \subset U_h$

**Figure 5.5:** Diagram of the stability analysis for shared control: $U_r \cap U_h \neq \emptyset$, $U_r \not\subset U_h$ and $U_h \not\subset U_r$. 
4. $U_r \cap U_h = \emptyset$. This is an unstable situation. Here, the definition of stability is the robots not colliding with obstacles. The feasible set of human inputs and the feasible set of autonomous control inputs do not intersect. There is no value of the allocation weight that can help us achieve both goals (following the user and avoiding obstacles). This situation may occur when the user does not notice the danger or obstacles. This situation is the most dangerous one and may cause a collision. Therefore, we stop the movement of the robot ($u_s = 0$) immediately. Fig. 5.6 shows this situation.

### 5.3.3 Algorithm to Calculate the Allocation Weight

Based on the stability analysis of the adaptive shared controller, we develop an algorithm to calculate the allocation weight. From the stability analysis, we know that the upper and lower bounds of the allocation weight are important values to consider.
in the choice of an allocation weight. Because the feasible sets of each control inputs are convex, there is only one value on both the boundary of the feasible set $U_r$ or $U_h$ and the closed segment line of $u_s$. Therefore, we can use a bisection method to find $\alpha_{su}$ and $\alpha_{sl}$. The bisection method is a simple root-finding method belongs to the numerical method. This method repeatedly bisects a given interval and then selects a subinterval that has a root for further processing [93]. The pseudo code of the bisection method is shown in Algorithm 1.

**Algorithm 1** Bisection Method

**Inputs:** A given function $f(x)$, boundary values $a, b$, tolerance $tol$, maximum iterations $NMAX$

**Variables:** Current iterations number: $N$, Current midpoint: $c$

**Conditions:** $a < b$, either $f(a) < 0$ and $f(b) > 0$ or $f(a) > 0$ and $f(b) < 0$

**Output:** a value that may differ from the real root of $f(x) = 0$ by less than $tol$

$N \leftarrow 1$

while $N \leq NMAX$ do

$c \leftarrow (a + b)/2$

if $f(c) = 0$ or $(b - a)/2 < tol$ then

$c$ is the answer

Stop

end if

$N = N + 1$

if $\text{sign}(f(c)) = \text{sign}(f(a))$ then

$a \leftarrow c$

else

$b \leftarrow c$

end if

end while

For the bisection method, we can use tolerance and maximum iterations to control the accuracy of the result. If the tolerance is large and the maximum iteration is small, the accuracy of the final result is low and the computational burden is light. If the tolerance is small and the maximum iteration is large, the accuracy of the final result is high, but the computational burden is heavy. Therefore, we can utilize these two
parameters to control the accuracy of the result and the computational burden for
the computer. After receiving the result of $\alpha_{su}$ and $\alpha_{sl}$, we will calculate $\alpha_s$. The
simplest way to get $\alpha_s$ is to make

$$\alpha_s = \frac{1}{2}(\alpha_{su} + \alpha_{sl}). \quad (5.4)$$

(5.4) means that $\alpha_s$ lies in the stable range and also has a margin to guarantee
stability.

The pseudo code below helps illustrate the idea for this algorithm.

**Algorithm 2** Simple algorithm to calculate the allocation weight

```
find $\alpha_{su}$ such that $\dot{V}_r(u_s(\alpha_{su})) = 0$ using the bisection method
find $\alpha_{sl}$ such that $\dot{V}_h(u_s(\alpha_{sl})) = 0$ using the bisection method
if $\alpha_{su} > 1$ then
  $\alpha = 1$
end if
if $\alpha_{sl} < 0$ then
  $\alpha_{sl} = 0$
end if
return $\alpha_s = \frac{1}{2}(\alpha_{su} + \alpha_{sl})$
```

### 5.4 Summary and Conclusion

In this chapter, we present the design of an adaptive servo level shared controller for
the mobile assistive robot. This shared controller is used to dynamically assign the
control weight to the human and to the autonomous control, according to the real-
time situation. The stability of this shared controller in different situations is analyzed
by a convex analysis. Four different situations are discussed in detail. Based on this
stability analysis, a simple algorithm to calculate the allocation weight is developed
to calculate a stable shared control output.
Chapter 6

Optimization for Adaptive Servo Level
Shared Control

6.1 Introduction

In Chapter 5, a simple algorithm to calculate $\alpha_s$ is given based on the stability analysis. This calculation result could give us a satisfying value of $\alpha_s$ that would enable us to satisfy both objectives (track the users movements and avoid obstacles). However, for the proposed mobile assistive robot, we need to know whether there is an optimal solution for $\alpha_s$ to get the best use of human and artificial intelligence. This optimal solution can help the robot find the shortest route to finish the assistive task, and the fastest route to move out of potential danger.

In this chapter, we introduce a method to solve the optimal shared control problem for our assistive robot. The basic idea is to model the optimal shared control as a multi-objective optimization problem. We can then solve this problem to obtain the optimal shared control output.

The organization of this chapter is as follows. First, we introduce some basic concepts of the multi-objective optimization. Afterwards, we analyze our optimal shared
control problem to find the objective functions. After this analysis, we model the optimal shared control problem as a nonlinear constrained multi-objective optimization. Finally, we find a suitable way to solve our problem.

6.2 Introduction to the Multi-objective Optimization

Based on the stability analysis of the adaptive shared controller, we can see that the controller in the first and the fourth situations are simple. In the first situation, the user controls the movements of the robot perfectly. In the fourth situation, the robot is stopped because of the users dangerous control input.

However, in the second and the third situations, the control input calculation is more challenging. In these situations, the robot needs to complete two objectives. One objective is to track the users movement, while the other is to avoid obstacles. Based on our analysis, the user deficiency causes these two objectives to contradict each other. If the robot follows closely behind the user, it may collide with the obstacle. If the robot avoids the obstacle, the robot may lose track of the users movements. This contradictory feature of the objectives leads us to model the shared control problem as a multi-objective optimization problem.

Multi-objective optimization can also be called multi-objective programming, or multi-criteria optimization; it is to simultaneously optimize two or more objectives that are conflicting with each other subject to certain constraints [94] [95]. The mathematical expression for multi-objective optimization is:

\[
\begin{align*}
\text{Minimize} & \quad f_1(x), f_2(x), \ldots, f_N(x) \\
\text{Subject to} & \quad g(x) < 0, h(x) = 0, x_l < x < x_u
\end{align*}
\]
Here, $f_1(x), f_2(x), ..., f_N(x)$ are the objective functions. $x$ is the variable, $g(x)$ is the function for the inequality constraint, $h(x)$ is the function for the equality constraint, and $x_l$ and $x_u$ are the lower and upper limit of $x$.

The nature of multi-objective optimization is that the objective functions in the problem are in conflict with each other. Tradeoffs always exist in the optimizing process of the problem. While optimizing the problem, the decision maker may have the following experience: When one objective function improves, the other objective function regresses. During this optimizing process, the decision maker has to accept a suboptimal cost value for single objective functions. In the final result, the decision maker will find the following situation: When improving an objective function based on this solution, other objective functions will be worse.

### 6.3 Multi-objective Optimization for Shared Control Problem

To model the shared control problem in our assistive robot, we need to determine two important things. One is the objective functions of the problem. The other is the constraints of the variables in the problem. In this section, we will start by providing the objective functions. We will then give the constraints of the variables in these functions.

In our assistive robotics system, the robot has two tasks: following the users movements and avoiding obstacles. Therefore, these two tasks can be treated as two objectives in a multi-objective optimization problem. The Lyapunov functions in the controller design process can be the potential objective functions, because when we minimize these functions, we also achieve the two objectives. However, if we use these Lyapunov functions as objective functions unchanged, it increases the computational
CHAPTER 6. OPTIMIZATION FOR ADAPTIVE SERVO LEVEL SHARED CONTROL

To simplify the computation, we define two new objective functions: For the tracking controller, we use \( f_1 = \frac{1}{2} r_e^2 + \frac{1}{2} \theta_e^2 \) as the objective function. The first term is the position error. If this error decreases, the robot and the user will get closer to each other. The second term is the orientation error. If this error decreases, the robot and the user will point in the same direction. For the obstacle avoidance algorithm, we use \( f_2 = \frac{1}{2} I_D^2 \) as the objective function. If the intrusion ratio \( I_D \) decreases, the robot will stay away from the obstacles.

Although these two objective functions are simplified, we can still achieve the two basic objectives of the robot. The variable of the optimization problem is the allocation weight \( \alpha_s \). This variable defines the value of the shared controller \( u_s \) by (1.1). Based on \( u_s \), the two objective functions \( f_1 = \frac{1}{2} r_e^2 + \frac{1}{2} \theta_e^2 \) and \( f_2 = \frac{1}{2} I_D^2 \) can be calculated through the kinematics model of the mobile robot.

The variable of the optimization problem is the allocation weights which gives the control input of the mobile robot. As seen in the stability analysis in Chapter 5, the shared controller in the first and the fourth situation is simple. We need only put \( \alpha_s = 0 \) for the first situation and stop the robot in the fourth situation.

In the second and third situations of the stability analysis, the allocation weight \( \alpha_s \) has constraints. In the second situation, \( \alpha_{s_l} < \alpha_s < 1 \). In the third situation, \( \alpha_{s_l} < \alpha_s < \alpha_{s_u} \). In these situations, the feasible set of \( u_s \) provides a constrained range for the solutions of the problem. \( \alpha_{s_l} \) and \( \alpha_{s_u} \) are the upper and lower bounds for \( \alpha_s \), which also gives the control input for the robot. Therefore, the original shared control problem converts to a constrained multi-objective optimization problem.

With the objective functions and variable constraints, we can model this optimization problem as:

\[
\text{Minimize } f_1(\alpha_s), f_2(\alpha_s)
\]
CHAPTER 6. OPTIMIZATION FOR ADAPTIVE SERVO LEVEL SHARED CONTROL

Subject to $\alpha_{s_l} < \alpha_s < \alpha_{su}$

Here, $f_1(\alpha_s) = \frac{1}{2}r_e^2 + \frac{1}{2}\theta_e^2 = \frac{1}{2}f_{r_e}(\alpha_s) + \frac{1}{2}f_{\theta_e}(\alpha_s)$ and $f_2(\alpha_s) = \frac{1}{2}I_D^2 = \frac{1}{2}f_{I_D}(\alpha_s)$. $f_{r_e}, f_{\theta_e}$ and $f_{I_D}$ are the functions that calculate the objective function from $\alpha_s$. The calculation steps are the same for these three functions. First, we use $\alpha_s$ and the shared control equation to calculate the control input of the robot. Then, we use this control input, the kinematics model of the mobile robot, and definitions of these objective functions to calculate the value of these objective functions.

This problem has one variable $\alpha_s$ and two quadratic objective functions. For a standard multi-objective optimization problem, we need to solve all of the solutions. The common methods of solving multi-objective optimization problems include classical methods such as weighted-sum, weighted-metric, and evolutionary algorithms such as genetic algorithms [96]. These algorithms are developed based on different design requirements. Therefore, we need to find a suitable calculation method to solve our multi-objective optimization problem for the shared controller.

There are several factors that need to be considered while designing the algorithm:

1. The calculation time for the controller is limited. The status and control signal for the mobile robot are updated approximately every 0.01sec. This time is fixed since the data transmits through a serial port between the robot and the computer. The data transmission rate of the serial port is 52kbps. One data package of the robots status and control signal transmits through this serial port in about 0.01 sec. For most of the multi-objective optimization algorithms, the calculation process is time consuming. Therefore, this 0.01 sec is a difficult constraint to implement in an algorithm for our optimization problem.

2. It is not necessary to solve all of the solutions. In our assistive robot system, the shared controller should have a decided solution for every time step, in order
to properly command the robot. Otherwise, the robot will wait for the user to choose the solution at every time step (100 times/sec), which is an impossible action for a human. Therefore, we do not need to calculate every solution but just one suitable solution.

3. The objective function and the constraints are nonlinear. The two objective functions are two quadratic functions which are nonlinear functions. These objective functions are calculated from the kinematics model of the mobile robot which also contains nonlinear functions. Therefore, these nonlinear functions make the computation process even harder.

In all the factors mentioned above, the most important one is the time limitation. As a real-time system, all autonomous algorithms such as the control algorithm, the optimization algorithm, and the velocity estimation algorithm should be finished within the sample time. Otherwise, the robot cannot move as desired. Therefore, we need to develop an algorithm that can complete all the steps within the time constraint. Moreover, if the optimal solution is difficult to obtain, the suboptimal solution is acceptable for this optimization problem.

6.4 Design of the Optimization Algorithm for the Shared Controller

6.4.1 Analysis of the Optimization Problem

Based on the aforementioned considerations for designing the optimization algorithm, the first step is to redefine the original multi-objective optimization problem into a new optimization problem.
In the classical method of the multi-objective optimization problem, the first step is to convert the original multi-objective optimization problem to a single objective optimization problem. This conversion is based on some predefined criteria. One approach is multiplying each objective function by a weight and summing up these results to form a new single objective function. This method is called weighted-sum method. In this method, the choice of the weight values is very important. Usually, the weights are given by the user, according to the specific circumstances. For example, if we are dealing with the two objectives of minimizing the wasted material of manufactured products and minimizing the cost of the products, a possible solution is to minimize a weighted sum of these two objectives. If we are concerned with the environmental impact, we might put more weight on minimizing the wasted material. If we want to save money while manufacturing these products, we might put more weight on minimizing the costs.

Inspired by the weighted-sum method for multi-objective optimizations, we apply the same concept to our optimization problem and assign weights to each objective function. The choice of the weights values, however, is another significant problem. In our system, these weights are chosen based on a safety measure. Because assistive robot systems target group of people are senior and disabled people, the system should ensure the safe operation of the robot and the protection of the user. In addition, in the latest interpretation of Asimovs three laws of robotics from [97]:

1. “A human may not deploy a robot without the human-robot work system meeting the highest legal and professional standards of safety and ethics.

2. A robot must respond to humans as appropriate for their roles.

3. A robot must be endowed with sufficient situated autonomy to protect its own existence as long as such protection provides a smooth transfer of control that
These three laws can be applied in our assistive robot system. For the first law, we know that the user cannot put the robot in a dangerous situation. However, since the users of our system are senior and disabled people, their low level of cognitive and sensory abilities may not enable them to always keep the robot safe. For example, the user may accidently command the robot to collide with an obstacle. According to the second law, while the system is operating, the robots main concern should be the safety of the whole system. Therefore, the robot itself should be able to help the user stay safe. When designing the optimization algorithm, the safety of the user is the top priority. There are two dangerous situations in our system:

1. The robot and user’s position errors are too large.

2. The robot may collide with obstacles and the user may get hurt by this collision.

For the first situation, the larger position error will be constrained by the safety control algorithm developed in Chapter 3. For the second situation, the robot will be guided by the shared controller to avoid obstacles.

To solve the optimization shared control design problem, we need to convert the multi-objective optimization problem to a new optimization problem. With the safe operation of the system as the top consideration, the aforementioned weighted-sum concept is applied to convert this multi-objective problem to a single objective optimization problem. The procedure is simple. First, we multiply each objective function of the original multi-objective optimization problem by some online adaptive weights. These weights are given by the assistive robot systems safety situation in real-time. Then, we calculate the sum of these objective functions to form a single objective function. Because the variable of the optimization problem is still $\alpha_s$, the constraints of $\alpha_s$ in the converted optimization problem are the same as in the original one.
For the online adaptive weights, we chose the amount of obstacles as a safety indicator. If there are a lot of obstacles in the surrounding area, the system is in a more dangerous situation. We can use the intrusion ratio $I_D$ in Chapter 3 as the indicator.

$$I_D = \int_{-\pi}^{\pi} \frac{d_h(\alpha_D) - d(\alpha_D)}{d_h(\alpha_D)} d\alpha_D. \quad (6.1)$$

Now, the new single objective optimization problem can be expressed as:

Minimize $(1 - I_D)f_1(\alpha_s) + I_Df_2(\alpha_s)$

Subject to $\alpha_{sl} < \alpha_s < \alpha_{su}$

In this new objection function, when there are more obstacles, $I_D$ will increase. $f_2(\alpha_s)$ will play a more important role in the total objective function. This means that the robot will put the avoidance of obstacles as an important objective. When less obstacles are presents, $I_D$ will decrease. $f_1(\alpha_s)$ will play a more important role in the total objective function. This means that the robot will put following the user’s movements as an important objective.

### 6.4.2 Normalization of the Objective Functions

In Section 6.4.1, each objective function was assigned an online adaptive weight by the computer, based on a real-time situation. However, the magnitude of the original objective functions may be different because of their different calculation method. This magnitude difference may cause different weights on the objective functions. The total objective function may be biased toward one objective function because of this magnitude difference. This is an unwanted phenomenon that destroys the balance of the objectives. Therefore, we should consider this magnitude difference
and normalize each objective function to the same magnitude to avoid the unbalance of the different objective functions.

In our assistive robot system, the two different objective functions are based on tracking errors and the distance of obstacles. The tracking errors are calculated based on the summary of two different objectives: the position difference between the Phantom Omni and the robot, and the orientation difference between the Phantom Omni and the robot. The distance of the obstacles is calculated based on the robot and the nearest obstacle. The largest distance between the Phantom Omni and the robot is about 20cm, and the largest orientation difference between the Phantom Omni and the robot is about 90 degree. However, the distance between obstacles and the robot can be 50cm and have a larger magnitude than the tracking errors. Therefore, we need to normalize these two objective functions to the same magnitude.

Before giving different approaches for normalization, we introduce a normalization operator $Nor()$, and a normalization factor $\theta_i$. The normalization operator computes the normalized values of the objective functions. The normalization factor is the calculation factor. For example, we normalize the objective function $f_i$ by multiplying it by a number.

$$Nor(f_i) = \theta_i f_i.$$ \hspace{1cm} (6.2)

In this equation, $f_i$ is the objective function. $\theta_i$ is the normalization factor. We can also use a more complex method to calculate the normalization value.

There are three possible normalization approaches:

1. Multiply the objective function by its initial value $f_i(x_0)$. Here, $\theta_i = \frac{1}{f_i(x_0)}$.

2. Multiply the objective function by its maximum or minimum value $f_{max/min}(x)$. Here, $\theta_i = \frac{1}{f_{max/min}(x_0)}$. 
3. Using the equation

\[ Nor(f) = \frac{f - f_{\text{min}}}{f_{\text{max}} - f_{\text{min}}} \]  

(6.3)

This equation gives us the relative value of the objective function comparing it with the difference of \( f_{\text{max}} \) and \( f_{\text{min}} \).

For the first two approaches, it is very hard to get a usable or constant value for the normalization factor. If we apply the first approach in our system, the initial value of the objective function is zero. This value cannot be taken as a normalization factor as we would obtain an infinite normalized value of the objective function. If we apply the second approach in our system, the normalized value of the objective function is also different from case to case because of the randomly placed obstacles.

The third approach is a practical way of normalizing the objective function in our system. In this approach, the difference in the calculated maximum and minimum values prevents the possible bias that can occur when using a single maximum or minimum value, as in the second approach. We initially conduct some experiments to get values of \( f_{\text{max}}() \) and \( f_{\text{min}}() \). Since the surrounding environments of the robot varies from time to time, it is very hard to get an accurate value of the maximum and minimum value. We calculate the average values of \( f_{\text{max}}() \) and \( f_{\text{min}}() \). With these two average values and the normalization (6.3), we obtain normalized values of the objective functions.

Therefore, we can write the final form of our optimization problem as:

\[
\text{Minimize } I_D Nor(f_1(\alpha_s)) + I_D Nor(f_2(\alpha_s))
\]

\[
\text{Subject to } \alpha_{s_l} < \alpha_s < \alpha_{s_u}
\]
6.5 Solution to the Single Objective Optimization Problem

6.5.1 Interior Point Method for Nonlinear Optimization Problem

If we express our objective optimization problem in a formal mathematical format, it is called a constrained nonlinear optimization problem. In literature, the interior point method is a method commonly used to solve the constrained nonlinear optimization problem [95]. This method is an extension of Newtons method, which is widely used for unconstrained nonlinear optimization problems.

The interior point method uses the same idea as the Lagrange multiplier, which converts the constrained optimization problem to an unconstrained optimization problem. This conversion uses a variable to multiply the constriction, and adds this result to the original problem in order to form an unconstrained problem. This converted unconstrained problem is also called the dual problem of the original problem.

Here we give a brief introduction of the interior point method:

Given a constraint nonlinear optimization problem:

\[
\text{Minimize } f(x) \\
\text{Subject to } g(x) < 0
\]

The introduced conversion idea is applied to this problem. We obtain a barrier function as:

\[
B(x, u) = f(x) - \mu \cdot g(x). \quad (6.4)
\]

The gradient of \( B(x, u) \) is
∇B(x, u) = ∇f(x) − μ \sum \frac{1}{g(x)} \nabla g(x). \hspace{1cm} (6.5)

To find the minimum value of (6.4), Newton’s method is applied. This method has an iterated process. We need to update an iterated equation at every step to find a final solution. The iterated equation is

\[ x_0 = x + \alpha \Delta x. \] \hspace{1cm} (6.6)

The update information \( \Delta x \) can be solved by Karush-Kuhn-Tucker (KKT) system, which is

\[
\begin{pmatrix}
H(f) & -J(g)^T \\
\Lambda J(g) & C
\end{pmatrix}
\begin{pmatrix}
\Delta x \\
\Delta \lambda
\end{pmatrix}
=
\begin{pmatrix}
-\nabla f + J(g)^T \lambda \\
u - c\lambda
\end{pmatrix}. \hspace{1cm} (6.7)

Here \( H(f) \) is the Hessian matrix of \( f(x) \), \( J(g) \) is the Jacobian matrix of \( g(x) \), \( \lambda \) is a slack variable that satisfies \( \nabla f(x) - J(g)\lambda = 0 \), \( \Lambda \) is a diagonal matrix of \( \lambda \), and \( C \) is the constant in the matrix.

The interior point method is a multiple-stage and time consuming method. Some software packages are developed to implement this method. For example, IPOPT [98] is an open-source software package for large-scale nonlinear optimization using the interior point method. However, the nonlinearity and numerical precision make the solving process very time and resource consuming. A problem with the PDE-constrained optimization problem introduced in [99] needs an 8-core Xeon Linux workstation to calculate for 48 hours. In our robotic system, we do not have such computational power, which is why we need to develop a more practical method of solving the optimization problem.
6.5.2 Equal Interval Point Method

With the constraints of our system’s hardware (notebook’s computational ability, sensor’s resolution, etc.), an equal interval point method is proposed to solve the optimization problem. The main idea of this method is to divide the whole possible range of the variable $\alpha_s$ into equal intervals. We then pick up these possible $\alpha_s$ with these equal intervals to form a possible set for $\alpha_s$. We find an $\alpha_s$ that can get the minimum value of the objective function. This $\alpha_s$ is the final allocation weight value for the shared controller.

The detailed procedure of this method is listed below:

1. Generate a feasible allocation weight set $A_s$ by the following equation:

$$A_s = \{\alpha_s | \alpha_s = \alpha_l + \frac{i}{N} (\alpha_{su} - \alpha_{sl})(i = 1, 2, 3...N)\} \quad (6.8)$$

2. Calculate the shared control outputs $u_s$ (the control input for the robot) for the entire feasible $A_s$ set based on the shared control equation $u_s = (1 - \alpha_s)u_h + \alpha_s u_r$.

3. Calculate the objective functions by different $u_s$ and use the normalization operator $Nor()$ to normalize these objective functions.

4. Use the weighted-sum method to calculate the current objective functions for the system based on the different normalized objective functions in step 3.

5. Find the minimum system’s objective function and use its corresponding shared control output $u_s$ as the final control command for the robot.

Let us consider these procedures in more detail. In the first step, the feasible allocation weights $\alpha_u$ are generated from (6.8). In this equation, we divide the feasible range of $\alpha_s$ between $\alpha_{su}$ and $\alpha_{sl}$ into N equal intervals. $\frac{\alpha_{su} - \alpha_{sl}}{N}$ gives the value of this
interval. \( \alpha_s = \alpha_{sl} + \frac{\alpha_{su} - \alpha_{sl}}{N} \) gives each feasible value of \( \alpha_s \). The number of the interval \( N \) can be changed by the computational power of the specific system. These points between \( \alpha_{su} \) and \( \alpha_{sl} \) can be seen in Fig. 6.1. In steps 2-4, the control inputs of the robot and the normalized objective function are calculated for every feasible control input. Finally, from these objective functions, we find a minimum function and its corresponding control input as the command for the robot.

In this calculation process of the algorithm, we can see that the final result may not be the optimal value for the optimization problem, since our optimization problem has some nonlinear properties:

- The nonlinearity of the kinematics model of the mobile robot
- The objective function is a quadratic function

Moreover, the calculating time constraint of the system is 0.01sec. All these conditions make the problem very hard to solve. Therefore, a suboptimal solution is acceptable for the real time application. In Chapter 7, we will conduct a comparison experiment to prove that this optimization algorithm improves the performance of the system.

The pseudo code of this equal interval point method is shown in Algorithm 3
Algorithm 3 Equal interval point method to calculate the allocation weight

Generate an array for $\alpha_s$ as $\alpha_s[i] \leftarrow \alpha_l + i \frac{\alpha_{su} - \alpha_{sl}}{N} (i = 1, 2, 3...N)$

for $i = 1 \rightarrow N$ do

1. Calculate the shared control output $u_s = u_h + (1 - \alpha_s[i])u_r$
2. Calculate the normalized objective function $Nor(f_1(\alpha_s[i])) + IDNor(f_2(\alpha_s[i]))$.

end for

Find the minimum objective function of $Nor(f_1(\alpha_s[i])) + IDNor(f_2(\alpha_s[i]))$

Find corresponding $\alpha_s$ and $u_s$ for the minimum objective function
return $\alpha_s$

6.6 Summary of the Optimal Adaptive Servo Level Shared Control Algorithm

Combining the shared control algorithms from Chapter 5 and 6, we can get the whole optimal adaptive servo level shared control algorithm as Algorithm 4.

Algorithm 4 Optimal adaptive servo level shared control algorithm

Find $\alpha_{su}$ such that $\dot{V}_r(u_s(\alpha_{su})) = 0$ using bisection method
Find $\alpha_{sl}$ such that $\dot{V}_h(u_s(\alpha_{sl})) = 0$ using bisection method
if $\alpha_{su} > 1$ then
$\alpha = 1$
end if
if $\alpha_{sl} < 0$ then
$\alpha_{sl} = 0$
end if
Generate an array for $\alpha_s$ as $\alpha_s[i] \leftarrow \alpha_l + i \frac{\alpha_{su} - \alpha_{sl}}{N} (i = 1, 2, 3...N)$

for $i = 1 \rightarrow N$ do

1. Calculate the shared control output $u_s = u_h + (1 - \alpha_s[i])u_r$
2. Calculate the normalized objective function $Nor(f_1(\alpha_s[i])) + IDNor(f_2(\alpha_s[i]))$.

end for

Find the minimum objective function of $Nor(f_1(\alpha_s[i])) + IDNor(f_2(\alpha_s[i]))$

Find corresponding $\alpha_s$ for the minimum objective function
return $\alpha_s$

The first step in Algorithm 4 is the calculation of the boundary values of $\alpha_s$ which
are \( \alpha_{sl} \) and \( \alpha_{su} \) based on the different situations that are analyzed in Chapter 5. The bisection method is used in the calculation of the boundary values. By using these boundary values, the equal interval point method is applied to calculate the optimal value of \( \alpha_s \). Finally, we can use this \( \alpha_s \) to get \( u_s \) as the shared controller’s output.

### 6.7 Haptic Rendering Algorithm

A simple haptic rendering algorithm was introduced in Chapter 3. However, this rendering algorithm is only used for the tracking controller. For the shared controller, we need another haptic rendering algorithm. As mentioned in the introduction, this kind of shared control with a haptic rendering algorithm is called haptic shared control (HSC). In our application, because the amount of haptic feedback varies due to real situations, the HSC is a variable authority HSC (VAHSC).

Based on the aforementioned stability and multi-objective optimization analysis, a haptic rendering algorithm of VAHSC is developed. The haptic feedback is given based on the difference between the human control input and the shared control output, which is shown below:

\[
f_{va} = k_{va} |u_s - u_h|
\] (6.9)

With this feedback force \( f_{va} \), the operator could understand the intention of the robot and know what the difference is between both commands. Then, the operator could correct his command. Because the aforementioned stability analysis is just a generalized analysis, the design of the haptic feedback is case specific for the actual value of \( u_s \) and \( u_h \).

In our mobile assistive robot, the specific design of the haptic rendering algorithm is based on \( \Delta v = v_m - v_o \) and \( \Delta w = w_m - w_o \), which is shown below.
Here, $f_x$ and $f_y$ are the feedback forces along the $X$-axis and the $Y$-axis respectively, and $k_{f_v}$ and $k_{f_w}$ are gains to adjust the feedback forces.

The idea of this haptic rendering algorithm is illustrated in Fig. 6.2. The haptic feedback force points from $u_h$ to $u_s$, which means that the human command should be close to this shared control output. This shared control output is calculated as the optimal one for the mobile assistive robot, based on the current situation. If the user’s commands are close to this control output, the robot could quickly move out of the dangerous situation. For our specific application, (6.10) maps this feedback force to the haptic device.

\begin{align*}
\begin{cases}
    f_x &= k_{f_v} \Delta v \cos(k_{f_w} \Delta w), \\
    f_y &= k_{f_v} \Delta v \sin(k_{f_w} \Delta w).
\end{cases}
\end{align*}

**Figure 6.2:** Illustration of the haptic feedback force $f_{va}$ for VAHSC
6.8 Summary and Conclusion

In this chapter, we discuss the optimization for the shared control problem in our assistive robot. Since our shared control problem has the same intrinsic contradictory feature as the multi-objective optimization problem, we model our shared control problem as a multi-objective optimization problem. The analysis of the problem and details of the calculation algorithm are present in this chapter. At the end of this chapter, the whole optimal shared control algorithm is presented. In addition, based on this algorithm, a haptic rendering algorithm of VAHSC is presented.
Chapter 7

Experimental Results and Analysis

The overview of the mobile assistive robot system, the mobile robots controllers, the obstacle avoidance algorithm, and the shared controllers were introduced and presented in the previous chapters. The experimental results are an important assessment of the effectiveness of the developed system and algorithms. In this chapter, the experiment setup, the results analysis, and the comparison and discussion will be presented.

First, we will conduct experiments to test the performance of the two tracking controllers. Then, based on the comparison of the results, we will choose the better performing tracking controller for the mobile robot. In these experiments, we also tested the performance of the velocity estimation using the fuzzy logic technique and the safety control algorithm. Second, we will conduct comparison experiments between different shared control algorithms. Based on these experimental results, we will choose the better performing shared controller for the mobile assistive robot. Then, we conduct different navigation experiments in simple situations to test the performance of the mobile robot. Finally, we conduct experiments for the mobile assistive robot in a complex environment to test the overall performance of the shared controller and haptic feedback.
7.1 Experiments for the Different Tracking Controllers of the Mobile Robot

In Chapter 2, two kinds of nonlinear controllers for the nonholonomic mobile robot are developed. From the control theory perspective, the experiments are designed to observe the input and output signals of the system. In order to test the performance of these two controllers, two kinds of experiments are carried out:

- The designed trajectory experiments
- The user input experiments

In the first experiment, the mobile robot follows two kinds of designed trajectories. One trajectory is the straight line, set to $45^\circ$ to the front of the mobile robot. The other trajectory is the straight line set to $90^\circ$ to the front of the mobile robot. This experiment is to test the performance (such as overshoot, regulation time, etc.) of the controllers without human intervention.

In the second experiment, the mobile robot follows the users input. When the user moves the Phantom Omni stylus, the mobile robot follows the users movement. This is the same situation as the application of the proposed mobile assistive robot. The goals of this experiment are to test the performance of the controller under human control, the novel velocity estimation method, and the haptic feedback algorithm. A situation where the robot encounters obstacles is not considered at this time. Therefore, no obstacles are placed in the environment.
7.1.1 Experiments for the Rectangular Coordinate Controller

In Chapter 3, the first nonlinear controller of the nonholonomic mobile robot is derived in rectangular coordinate by the backstepping technique. The corresponding experiments are carried out as follows:

Results of Designed Trajectory Experiment

For the first designed trajectory, the robot has to follow a straight line with a velocity of 141 mm/s and a direction of 45° to the heading of the robot. For the second designed trajectory, this line is changed to a velocity of 100 mm/s and a direction of 90° to the heading of the robot, which is vertical to the initial position of the robot. Because there is no noise in the input signal for the designed trajectories, in order to verify the efficiency of the control algorithm, the velocity estimation algorithm is not applied in these two experiments. The results of the experiments are shown in Figs. 7.1 and 7.2.

From these experimental results, we can conclude that for a given trajectory, using the rectangular coordinate controller, the mobile robot is able to follow the position and orientation (Figs. 7.1(a) and (b) and 7.2(a) and (b)) and the velocity of the trajectory (Figs. 7.1(c) and 7.2(c)). Since the robot is a nonholonomic mobile robot, in order to follow a trajectory with a direction that is different from its position, the robot needs to turn and move forward at the same time. Therefore, there is a position error that appears in the results at the beginning of each experiment (Figs. 7.1(d) and 7.2(d)). The maximum position errors are around 180mm for a 90° trajectory and around 120mm for a 45° trajectory.
Figure 7.1: The designed trajectory experiment for the robot with the rectangular coordinate controller (45°)
Figure 7.2: The designed trajectory experiment for the robot with the rectangular coordinate controller (90°)
CHAPTER 7. EXPERIMENTAL RESULTS AND ANALYSIS

Results of the Human Input Experiment

The real human input experiment is carried out to evaluate the performance of the controller, the velocity estimation, and the haptic rendering algorithm. The human user holds the Phantom stylus firmly and slowly moves it. First, the user moves the stylus in the translational direction, then turns the stylus to the left. These two actions verify the translational and rotational performance of the system, respectively. Figs. 7.3, 7.4 and 7.5 show the results of the experiment.

From the results, we can conclude that the rectangular coordinate controller works well with the human input. In Fig. 7.3(a) and (d), the position error finally converges
CHAPTER 7. EXPERIMENTAL RESULTS AND ANALYSIS

Figure 7.4: The feedback force in the human input experiment with the rectangular coordinate controller

Figure 7.5: The changing window size of the proposed velocity estimation method
to zero. The results of the force feedback algorithm are shown in Fig. 7.4. The feedback force is directly proportional to the position error and points to the mobile robot. With this feedback force information, the human user is informed of the error between the robot and his/her position. If the user moves too fast and the robot cannot follow its movements, the feedback force will increase. Then, the user knows he/she moves too fast for the robot to follow and will slow down. The effectiveness of the fuzzy logic adaptive velocity estimation method is shown in Fig. 7.5. When the position error and its changing rate change rapidly (in 0-10s and 40-50s), the window size shifts correspondingly.

### 7.1.2 Experiments for the Polar Coordinate Controller

#### Results of the Designed Trajectory Experiment

For the first designed trajectory, we make the robot follow a straight line with a velocity of 141 mm/s and a direction of 45° to the heading of the robot. For the second designed trajectory, this line is changed to a velocity 100 mm/s and a direction of 90° to the heading of the robot, which is vertical to the initial position of the robot. These are the same trajectories as the rectangular coordinate controller. The velocity estimation algorithm is also not applied in these two experiments. The results of the experiments are shown in Figs. 7.6 and 7.7.

From these experimental results, we can conclude that for a designed trajectory, using the polar coordinate controller, the mobile robot is able to follow the position and orientation (Figs. 7.6(a) and (b) and 7.7(a) and (b)) and the velocity of the trajectory (Figs. 7.6(c) and 7.7(c)). In Figs. 7.6(d) and Figs. 7.7(d), the maximum position errors are around 90mm for a 45° trajectory and around 120mm for a
Figure 7.6: The designed trajectory experiment for the robot with the polar coordinate controller (45°)
Figure 7.7: The designed trajectory experiment for the robot with the polar coordinate controller (90°)
90° trajectory. These position errors are much smaller than the position errors of the rectangular coordinate controller. In Figs. 7.6(b,c) and Fig. 7.7(b,c), the response time of the polar coordinate controller for translational and rotational velocity are shorter than the rectangular coordinate controller. However, the overshoot and stable time are longer than the rectangular coordinate controller in Figs. 7.1 and 7.2.

**Results of Human Input Experiment**

Fig. 7.8 shows the results of the human input experiment for the polar coordinate controller. The user moves the stylus to control the mobile robot in order for it to move in a circle. From the results, we can conclude that the polar coordinate controller works well with human input. From Figs. 7.8(a) and (d), the position error finally converges to zero and the robot finally moves to the desired destination. In Fig. 7.9, the feedback force always guides the user to the mobile robot’s position. The effectiveness of the fuzzy logic adaptive velocity estimation method is shown in Fig. 7.10. When the position error and its changing rate change rapidly (in 0-10s and 40-50s), the window size shifts correspondingly.

**7.1.3 Comparison of the Rectangular and Polar Coordinate Controllers for the Mobile Robot**

From the results of the two kinds of controllers for the mobile robot, we can see that the performances of these controllers are quite different. The rectangular coordinate controller has a long response time and a large position error. Nonetheless, the results of this controllers experiments have the smooth trajectories. The polar coordinate controller has the faster response time and smaller position error, but the results of
Figure 7.8: Human input experiment with the polar coordinate controller
Figure 7.9: The feedback force in the free space moving

Figure 7.10: The changing window size of the proposed velocity estimation method
CHAPTER 7. EXPERIMENTAL RESULTS AND ANALYSIS

this controllers experiments have a noisy trajectory.

The different performance of the controller comes from the coordinate transformation of Phantom Omni. The encoders on the Phantom Omni record the degree value of each joint. In the rectangular coordinate, this record of encoder is transformed to the Cartesian coordinate. This transformation loses some precision and brings some noise to the raw data. Therefore, in order to cancel the noise in the data, the velocity estimation method for the rectangular coordinate controller gives a larger window size (mean value 308.4, variance 7.3) than the velocity estimation method for the polar coordinate controller (mean value 305.4, variance 3.6). With this larger window size, the output trajectory of the rectangular controller is smoother than the trajectory of the polar coordinate controller.

It is very hard to use both of the controllers together, which is why we have to choose the controller that is the most suitable for our application. Because the proposed assistive robot needs to interact with humans, the ideal controller for the robot should have features like a fast response time, a small position error, and smooth trajectories. For the wheelchair robot, human users sit on the wheelchair, so the robot should move smoothly to ensure quality. However, for our application, the human users are only connected to the robot by the Phantom Omni and stand beside the robot. Human users may not always require a smooth trajectory for their operation. Therefore, the smoothness is not as important compared to the other two features (fast response and small position) of the system. This analysis means that when we choose the controller, a fast response time and a small position error are the most important features. Therefore, the polar coordinate controller is chosen as the tracking controller for the mobile robot.
7.2 Experiments for the Safety Control Algorithm

In order to test the efficiency of the safety control, two kinds of experiments are conducted. In one, the user controls the robots movements without safe control. In the other, the user controls the robots movements with safe control. All the robots controllers for these two experiments are polar coordinate controllers.

First, the robot is moved without the safety control. The result is shown in Fig. 7.11 and Fig. 7.12. In Fig. 7.11 (a), we can see that there are spikes in the input from the human who is controlling the robot. As seen in Fig. 7.12, the base angle of the Phantom $\phi_e$ reaches its limit (flat part). The reason is that when the turning action is fast, the robot cannot turn as fast as the human user (the low rotational velocity shown in Fig. 7.12). This is a dangerous scenario that may damage the haptic devices and injure the human user if it happens in the real application.

Subsequently, we performed the experiment with the safety control algorithm. From Fig. 7.13 and Fig.7.14, we see that the Phantom base angle cannot reach its limit since the safety control algorithm makes the mobile robots rotational velocity high enough to prevent this from happen. The high rotational velocity can be observed as two peaks in Fig. 7.14.

7.3 Experiments for the Shared Control Algorithms

In Chapters 5 and 6, we developed two kinds of shared controllers. One controller is developed based on the average method. The other controller is developed based on
Figure 7.11: Robot moving without safety control algorithm
CHAPTER 7. EXPERIMENTAL RESULTS AND ANALYSIS

111

Figure 7.12: $\phi_e$ and rotational velocity (no safety control algorithm)

the multi-objective optimization method. First, we carry out experiments to compare the performance of these two shared controllers. Then, we choose the shared controller that performed the best. We will use this better performing controller to carry out experiments where the robot will navigate the environment. In addition, a comparison experiment between the shared controller with haptic feedback and without haptic feed is conducted to test the effectiveness of the haptic force feedback.

7.3.1 Comparison Experiment between the Two Shared Controllers

In Chapters 5 and 6, two kinds of shared controllers are developed based on different principles. We need to know which controller has a better performance and is more suitable for our application. Because this comparison experiment should be in the
Figure 7.13: Robot moving with safety control algorithm
same experimental setup, the trajectory should be exactly the same. A simulation experiment is carried out to compare the performance of these controllers. In this experiment, the robot runs towards its goal and simultaneously avoids a static obstacle. The experiment runs in MobileSim (simulation software for the mobile robot). The map used in the simulation is made by the Mapper3 software. The trajectory and the position of the obstacle are the same for the two shared controllers. The experimental results are shown in Figs. 7.15.

From the experimental results, we can see that both the shared controllers can help the robot move to its goal and avoid the obstacle. However, from Fig. 7.15 (b), we can see that the objective functions in the shared controller based on the multi-objective optimization method decreases faster than the shared controller based on
Figure 7.15: Comparison experiment between the two shared controllers
(a) comparison between robot trajectories (b) comparison between the value of the objective functions
the average method. This means that the shared controller based on the multi-objective optimization method can move out of danger faster and perform better. Thus, for our assistive robotic system, we will use the shared controller based on the multi-objective optimization method.

### 7.3.2 Navigation Experiments

The first set of experiments is conducted to test the performance of the shared controller in different simple navigation scenarios. The experimental environment is a plane surface with some arranged walls, obstacles, etc., which are built by paper boxes (shown in Fig. 7.16).

We conduct the following experiments:
1. Wall following

2. Door passing

3. Narrow corridor passing

These three experiments are simple actions required from the robot while it navigates. These simple actions can be combined to form more complex navigating actions. In other words, the complex navigating actions can be decomposed into these simple actions. Therefore, we first need to know the performance of the robot in these simple actions, in order to make the corresponding adjustments to the algorithms. After these simple action experiments, we also conduct the experiment in a complex environment to test the overall performance of the robot and haptic feedback.

**Wall Following Experiment**

In this experiment, the paper boxes form three connected walls, as shown in Fig. 7.17. When we start the experiment, the user holds the Phantom Omni stylus and controls the robot so that it moves along the wall. We notice that the robot cannot be very close to the wall. This is because of the effectiveness of the DVZ around the robot, as part of the obstacle avoidance algorithm. When the robot gets too close to the wall, which is a dangerous action, this virtual zone will remove the robot from the potential danger. When the robot goes straight and meets the wall on the right side, there is more obvious evidence of the effectiveness of the DVZ and the shared controller. Here, the user intends to command the robot to move toward the wall. The DVZ gives the avoidance command to the shared controller. The shared controller calculates the allocation weights $\alpha_s$ and gives the control command to the mobile robot. We can see that $\alpha_s$ is around 0.7 at this point (shown in Fig. 7.17(b)). The obstacle avoidance algorithm takes most of the control from the robot, and the robot therefore turns in a
safer direction immediately. When the robot meets the other wall, the same process happens. $\alpha_s$ increases to help the robot move out of danger. The dynamic process of the change on $\alpha_s$ can be found in Fig. 7.17(b). We also notice that there are some flat parts in $\alpha_s$. This phenomenon is caused by the numerical accuracy of Algorithm 1. Since the bisection method is involved in this algorithm, we need to adjust the maximum iterations and tolerance to meet the fixed updated rate of 100Hz. This gives less numerical accuracy to the shared control algorithm. Therefore, some flat parts exist in $\alpha_s$. This effect can be eliminated by upgrading the CPU of the notebook, which will be done in our future work.

**Narrow Corridor Passing Experiment**

Within this experiment, we perform two separate experiments with corridors of different widths. In the first experiment, the width is around 880mm. In the second experiment, the width is around 680mm. The translational velocity of the robot is faster in the second experiment. In both experiments, the robot can safely reach its destination. We also find an interesting phenomenon; by comparing these two experiments, we find that when the robot is moving faster in a narrower corridor, the robot has an unsmooth trajectory (Fig. 7.18(a) and Fig. 7.19(a)). The reason for this phenomenon is the change in the size of the DVZ. The DVZ enlarges when the robot moves faster. Thus, the robot interacts with the narrower corridor with a larger DVZ. This larger DVZ will give a higher value to the obstacle avoidance control command. The robot will turn faster with a higher rotational velocity. For this reason, the trajectory in the second experiment is unsmooth. This is the same phenomenon that occurs for humans; a humans trajectory is not straight when he moves very fast in a narrow corridor. Because of this phenomenon, the user should move slowly when
Figure 7.17: Wall following experiments (a) human and robot trajectory (b) $\alpha_s$ in the experiment
passing a narrow corridor, in order to avoid an unsmooth trajectory for the mobile robot.

**Door Passing Experiment**

In this experiment, the paper boxes form a door with a width of around 880mm. The initial position of the robot is parallel to the door. The user needs to command the robot to turn right and pass the door. We can see from Fig. 7.20 that the user turns the robot too late. The robot may collide with the right side of the door. At this time, $\alpha_s$ increases and the robot turns to avoid the collision. Then, the user safely commands the robot to move across the door.

**Comparison Experiment for the Haptic Feedback in a Complex Environment**

In this experiment, the user operates the robot in two different situations. The difference is that one has the haptic feedback and the other does not have the haptic feedback. The experimental environment is a complex plane surface with arranged walls, obstacles, etc., which are built by paper boxes (shown in Fig. 7.21).

Figs. 7.22 and 7.23 show the results of the experiments. In both situations, the user can operate the robot to reach their goals in a complex environment. Both the robot and the user safely pass the door, avoid the obstacle, follow the wall, and pass the narrow corridor. Finally, the robot stops when the user puts the stylus into its initial position.

In the second situation with haptic feedback, we can see that the mean value of
Figure 7.18: Narrow corridor passing experiment (door width: 880mm) (a) human and robot trajectory (b) $\alpha_s$ in the experiment
Figure 7.19: Narrow corridor passing experiment (door width: 680mm) (a) human and robot trajectory (b) $\alpha_s$ in the experiment
Figure 7.20: Door passing experiments (a) human and robot trajectory (b) $\alpha_s$ in the experiment
the allocation weight is smaller than the mean value in the second situation without haptic feedback (shown in Fig. 7.24). This is especially found in the later parts of the two trajectories. In these parts, the shared controllers are actively involved in the control of the robot (high value of \( \alpha_s \)). In the first situation, the VAHSC gives haptic feedback to the user to inform him when he acts poorly. The user can therefore adjust his actions appropriately. Thus, the shared controller could put less authority weight on \( u_r \) and more authority weight on \( u_h \) to let the user control the robot. For the first situation, there is no feedback from the robot to the user. If the user acts poorly and commands the robot into a dangerous situation, the robot cannot inform the user about the situation. Thus, the shared controller puts more authority weight on \( u_r \) and less authority weight on \( u_h \) to ensure the safety of the system. Based on these comparisons, there are two advantages to VAHSC: 1. The
Figure 7.22: Complex environment experiment without haptic feedback (a) human and robot trajectory (b) $\alpha_s$ in the experiment
Figure 7.23: Complex environment experiment with haptic feedback (a) human and robot trajectory (b) $\alpha_s$ in the experiment
mobile assistive robot could run more safely. 2. The user gains more control authority of the robot. Therefore, the shared controller with haptic feedback (VAHSC) has a better performance than the shared controller without haptic feedback.

7.3.3 Summary and conclusion

In this chapter many experiments are conducted to verify the control algorithm developed for the mobile assistive robot. After the experiment for the mobile robot controller, the polar coordinate controller is chosen because of its fast response and small error. The comparison experiments for the safety controller show that the user could safely operate the robot without colliding with the robot. For the shared controllers, we choose the multi-objective optimization based controller because of its better performance. The simple navigation experiments show the good performance of the shared controller. In addition, the haptic feedback could help the user know the robots intention. The robot and the user could safely navigate the complex
environment with the help of the haptic feedback.
Chapter 8

Conclusion and Future Work

8.1 Summary of the Contributions

Nowadays, the number of senior people in the world is increasing. They need healthcare assistance to live an independent life, which is an important index of the quality of modern life. Together, senior and disabled people, who also need special healthcare assistance, make the requirements for specialized assistance and medical attention an urgent issue. One possible way of solving this issue is with an assistive robot. As a semi-autonomous system, the assistive robot can help the target group of people when assistance tasks are needed. This thesis proposes a prototype of a novel mobile assistive robot that can help senior and disabled people move heavy objects. This assistive robot interacts with the user through a haptic channel, which compensates for senior and disabled peoples decreasing vision and hearing capabilities.

The contributions are summarized as follows:

- Two kinds of nonlinear controllers are designed and developed for the mobile robot. One controller is designed based on the rectangular coordinate system, and the other controller is designed based on the polar coordinate system. The comparison and analysis of the experiments for these two controllers show that
the polar coordinate controller is more suitable for our application. An adaptive
windowing velocity estimation based on fuzzy logic is proposed to solve the
velocity estimation problem in the controller. A safety control algorithm is
developed to prevent injuries to the human user and damage to the haptic
device.

- An optimal adaptive servo level shared controller is designed for the proposed
  mobile assistive robot. When the robot encounters obstacles, this shared con-
troller is activated. This shared controller adaptively allocates the best suitable
  weight to the control inputs, for the mobile robot. The stability analysis of this
  shared controller is done by a convex analysis and set theory.

- Two haptic rendering algorithms are designed to generate the haptic feedback
  based on different situations. With this haptic rendering algorithm, the user
  can interact with the mobile assistive robot and increase the whole performance
  of the system.

8.2 Future Work

The developed system is, however, only the first prototype toward a real-life assistive
robot for senior and disabled people. More research work should be done to complete
the system.

- Some of the hardware parts will be redesigned by up-to-date devices. A more
  powerful embedded system will replace the notebook to connect the robot and
  haptic device. The systems performance will be improved by these updates.
  The algorithm will run faster and the operation of the robot will be smoother.
More experiments will be conducted to investigate the performance of the mobile assistive robot. These experiments will involve real users of the assistive robot (senior and disabled people). In addition to this experiment, more experiments will be conducted with doctors in the different application scenarios.

To ensure and improve the safety of the developed robotic system, we propose two evaluation metrics in any future work:

- The distance between the robot and the obstacles
- The difference between the human user control input and the shared control output

The first metric focuses on the potential danger the obstacles could cause the robot. The smaller the distance is, the more danger there is for the robot. We could use this metric to track how much danger the robot is moving towards. The second metric focuses on how much control difference there is between the human user and the shared control output. Since the robot knows the obstacles that surround it more than the human user does, the shared control output can command the robot in a safe way. This metric can be used to improve the safe operation of the robot. We could analyze the record of this metric and turn the parameters based on the analysis to ensure the safety of the system.


List of References


Appendix A

Contribution during PhD studies

Journals


Conferences
