

Advanced Semi-Autonomous Wheelchair System for People with Disabilities

by

Van Anh Nguyen

Submitted to the Faculty of Engineering and Information Technology

in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

at the University of Technology, Sydney

Sydney, May 2014

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree, except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

Van Anh Nguyen

Sydney, May 2014

Acknowledgements

First of all, I would like to express my sincere gratitude and appreciation to my supervisor, Professor Hung Tan Nguyen, for giving me the valuable opportunity to work on the intelligent wheelchair project under his expert guidance. I have greatly benefited from his continuous support, including his knowledge and spirit.

I would also like to thank my co-supervisor, Dr Steven Su, for his valued advice throughout this project. And, I greatly appreciate Ms. Lesley Nguyen for her great assistance in editing this thesis.

Special thanks to my wife, Bich Lien Nguyen, for her great encouragement and support. She is my best friend, my soul mate and my family.

Finally, I would like to dedicate this thesis to my mother, my father and my little daughter who are always a source of inspiration in my life.

Van Anh Nguyen

Sydney, May 2014

Contents

List of Figures	VII
List of Tables.	XI
Abbreviations	XII
Abstract.....	XIII
Chapter 1 Introduction.....	1
1.1 Problem statement.....	1
1.2 Objectives of the thesis.....	3
1.3 Thesis contributions.....	5
1.4 The structure of the thesis.....	6
1.5 Publications related to the thesis	8
Chapter 2 Literature Review.....	10
2.1 Introduction.....	10
2.2 Intelligent Wheelchairs	11
2.2.1 The system.....	11
2.2.2 Requirements of the Intelligent Wheelchair Application.....	13

2.3	Intelligent Wheelchair System Review	15
2.4	Autonomous Navigation Algorithms.....	23
2.4.1	Route Program Paradigm.....	24
2.4.2	Learning Machine Paradigm.....	26
2.5	Shared Control Strategies for Intelligent Wheelchairs	28
2.6	Discussion.....	30

Chapter 3	Real-time Free space Determination Based on Bayesian Recursive Technique.....	33
3.1	Introduction.....	33
3.2	Advanced Instrumentation.....	34
3.2.1	The Intelligent Wheelchair Design.....	35
3.2.1.1	Invacare Roller M1 with DX System.....	35
3.2.1.2	The Wheelchair Hardware System.....	36
3.2.1.3	The DX Master Remote Modification	39
3.2.1.4	Multi-Threading Based Software Design	40
3.2.2	Sensors.....	43
3.2.2.1	Laser Range Finder URG-04LX.....	43
3.2.2.2	Ultrasonic Sensor SRF01s	45
3.2.2.3	Encoders	46
3.2.3	The Human – Machine Interfaces.....	49
3.2.3.1	Joysticks.....	50
3.2.3.2	Iphone/Ipad Interface	51
3.2.3.3	Head Movement Interface.....	52
3.2.3.4	Brain Computer Interface.....	53

3.3	Free Space Determination Using Bayesian Recursive Technique.....	55
3.3.1	Real-time Map.....	56
3.3.2	Free Space Determination Using Bayesian Recursive Technique..	60
3.4	Experimental Results.....	65
3.4.1	Initialization.....	66
3.4.1.1	URG-04LX Noise Measurement Modelling.....	66
3.4.1.2	Free Space Passing Capability Modelling	67
3.4.2	Experiment 1: Passing a narrow door.....	69
3.4.3	Experiment 2: Passing a narrow door with a moving person.....	73
3.5	Discussion.....	74
Chapter 4	Autonomous Navigation Using Bayesian Neural Networks.....	77
4.1	Introduction.....	77
4.2	Autonomous Navigation System Based on a Bayesian Neural Network.....	78
4.2.1	Wheelchair Control Signals.....	80
4.2.2	Artificial Neural Network for Autonomous Navigation.....	82
4.2.2.1	Pre-Processing Environmental Data.....	83
4.2.2.2	Feed Forward Neural Network Architecture.....	86
4.3	Bayesian Neural Network Training.....	88
4.3.1	Network learning.....	88
4.3.2	Levenberg-Marquardt Optimization Algorithm.....	90
4.3.3	Bayesian Framework for Feed-Forward Neural Networks.....	93
4.3.3.1	Prior Weight Distribution.....	94
4.3.3.2	Posterior Weight Distribution.....	94
4.3.3.3	Posterior Probability of the Hyper-parameters.....	96

4.3.3.4	Determination of the Hyper-parameters.....	97
4.3.3.5	Model Comparison.....	98
4.3.4	Practical Implementation of Bayesian Technique with Levenberg – Marquardt Optimization.....	99
4.4	Experimental Results.....	100
4.4.1	Data Acquisition Experiments.....	100
4.4.1.1	Data Acquisition Methodology.....	100
4.4.1.2	Collecting training data experiments.....	102
4.4.2	Bayesian Neural Network Training Results	104
4.4.3	Experiment 1: Navigating SAM in Unknown and Dynamic Environments.....	106
4.4.3.1	Experiment Description	106
4.4.3.2	The Trajectory Observation.....	107
4.4.3.3	The Comparison between the Neural Network Model Performances.....	109
4.4.4	Experiment 2: The door passing experiment	110
4.4.4.1	Experiment Description	110
4.4.4.2	Door Passing Results	111
4.5	Discussion.....	113
 Chapter 5 Shared Control Strategy Based on Bayesian Theory for a Semi- Autonomous Wheelchair.....		115
5.1	Introduction.....	115
5.2	Advanced Shared Control Architecture for Semi-autonomous Intelligent Wheelchair.....	117
5.3	The implementation of Advanced Bayesian Shared Control Strategy.....	120
5.3.1	Adaptive Shared Control Strategy Based on Bayesian Inference.....	120

5.3.2	Likelihood Distribution Modelling.....	124
5.3.2.1	The Transition Probability.....	124
5.3.2.2	The Measurement Probability.....	126
5.3.3	Bayesian Neural Network for Head Movement Detection.....	127
5.3.4	Shared Control Framework Implementation.....	131
5.4	Experiment Results.....	133
5.4.1	Experiment 1: Advanced Bayesian Shared Control Strategy.....	133
5.4.1.1	The Head Movement Detection.....	133
5.4.1.2	The SAM Performance with the adaptive shared control strategy.....	136
5.4.2	Experiment 2: Evaluation of the SAM intelligent wheelchair with a group of able - bodied individuals.....	142
5.4.2.1	Experimental Study.....	143
5.4.2.2	Record Data and Data Analysis.....	144
5.4.2.3	Results.....	146
5.5	Discussion.....	152
Chapter 6 Conclusion and Future Work.....		155
6.1	Conclusion.....	155
6.2	Future Work.....	158
Appendix A Power Wheelchairs.....		160
A.1	Overview of Power Wheelchairs	160
A.2	Power Wheelchair Components	162
A.2.1	Battery.....	162
A.2.2	Motors.....	164

A.2.3	Electric Motor Drive.....	165
A.2.4	Control Module	167
Appendix B	Software Code.....	171
Appendix C	Questionnaire for the evaluation experiments.....	233
Bibliography.....		237

List of Figures

2.1	The block functions of an intelligent wheelchair.....	12
2.2	Intelligent wheelchair system, University of Michigan (IEEE, 1999).....	15
2.3	Intelligent wheelchair system, Osaka University (IEEE, 2003).....	16
2.4	Intelligent Wheelchair System, University of Bremen.....	17
2.5	Intelligent wheelchair system, University of Pennsylvania (IEEE 2007).....	18
2.6	Intelligent wheelchair system, University of Zaragoza (IEEE 2009, 2010).....	19
2.7	Intelligent wheelchair system, University of Toronto (IEEE 2007).....	19
2.8	Intelligent wheelchair system, University of Leuven (ELSEVIER 2010).....	20
2.9	Intelligent wheelchair system, University of Pittsburgh (JRRD 2011).....	21
2.10	Intelligent wheelchair system (TIM), University of Technology, Sydney.....	22
2.11	Intelligent wheelchair system, University of Saitama (ELSEVIER 2012).....	23
3.1	Invacare Roller M1 and electrical control diagram.....	36
3.2	The intelligent wheelchair connection diagram	37
3.3	A prototype of the intelligent wheelchair	38
3.4	The Connection of the Onboard Computer to the DX Master Remote.....	40
3.5	SAM's five threads with automatic load balancing across multiple cores.....	42
3.6	A flow chart of the program for the laser URG-04LX.....	44
3.7	Connection of the Ultrasonic Sensor SRF01s to the System.....	45
3.8	A flow chart of the program for SRF01s.....	46

3.9	Connection of wheel encoders to the system.....	47
3.10	The Geometry of the SAM.....	48
3.11	A flow chart of the program for the encoders.....	49
3.12	The voltage generated by the joystick sensor with different joystick.....	50
3.13	The iphone/ipad interface for SAM.....	51
3.14	The head movement interface parts.....	53
3.15	The wireless EEG hardware interface.....	54
3.16	The brain computer interface for SAM.....	55
3.17	The visible field of the wheelchair.....	57
3.18	The visible field of the wheelchair map.....	58
3.19	The wheelchair with a narrow opening.....	61
3.20	The shared control framework using Bayesian Recursive algorithm.....	65
3.21	The generalized control probability function and experimental data.....	68
3.22	SAM passing through a narrow free space.....	69
3.23	The states of the door are recursively updated under the Bayesian Framework.....	72
3.24	SAM passing through a narrow free space with a moving person.....	73
3.25	The states of the door are recursively updated with a moving person	74
4.1	Autonomous Control Function Diagram for SAM wheelchair	79
4.2	UTS Virtual Joystick Module.....	80
4.3	The UTS Virtual Joystick Module inputs and range of the signals.....	81
4.4	Steps to transform URG-04LX data.....	85
4.5	Feed-forward neural network structure for obstacle avoidance.....	87
4.6	Neural network training model for obstacle avoidance task.....	89
4.7	Estimation of log evidence for neural networks with different hidden nodes.....	105

4.8	Each neural network structure is asked to autonomously navigate the wheelchair following the blue line.....	106
4.9	An example of a test run conducted by a neural network with 4 hidden nodes.....	108
4.10	The door passing task is to assess the performance of the neural network with different structures.....	111
4.11	Results of the performance of the neural network models for the door passing task..	112
5.1	The Shared control architecture of the SAM intelligent wheelchair.....	118
5.2	The Bayesian network that SAM uses to estimate the most probable shared control actions.....	122
5.3	Head movement sensor outputs in response to forward nod and backward nod.....	128
5.4	Head movement sensor outputs in response to left nod and right nod.....	129
5.5	Feed-forward neural network for the head movement detection.....	130
5.6	Bayesian Shared Control Framework for the SAM Intelligent Wheelchair.....	132
5.7	Command patterns of the first user.....	134
5.8	Log Evidence versus number of hidden nodes.....	135
5.9	The Wheelchair trajectory when using the Bayesian Recursive Technique.....	137
5.10	The scene and real-time program interface around point A.....	138
5.11	The evidence values at the point A when SAM started.....	139
5.12	The scene and real-time program interface around point B.....	140
5.13	The action probabilities at the point B.....	140
5.14	The action probabilities at the point C.....	141
5.15	The action probabilities at the point D.....	141
5.16	The action probabilities at the point E.....	142
5.17	Designed course task for the evaluation experiment.....	144
5.18	Sample trajectories taken by the users.....	146

5.19	The box plot of quantitative performance results.....	149
5.20	The box plot of qualitative performance results.....	151
A.1	Power wheelchairs with conventional frame.....	161
A.2	Power wheelchairs with power base.....	162
A.3	Wheelchair gear-motors.....	164
A.4	Block Diagram of an Electric Motor Drive.....	165
A.5	Bi-direction PMBDC Motor H-Bridge Circuit.....	166
A.6	Hand held programmer setup and parameter samples.....	168
A.7	Wizard 5 setup and program parameters for DX master.....	169
A.8	HPP Plug-in setup.....	170

List of Tables

3.1	The performance of passing a door of the SAM wheelchair	68
4.1	Patterns collected from the acquisition data experiments.....	104
4.2	The results of the performance of the neural network with 3,4,5 and 6 hidden nodes when controlling wheelchair in ten trials	109
5.1	A number of command patterns of the five wheelchair users.....	134
5.2	Confusion matrix of the trained network with 3 hidden nodes.....	136
5.3	Confusion matrix of the trained network with 4 hidden nodes	136
5.4	The quantitative performance results (mean $\pm$ standard deviation).....	148
5.5	The qualitative performance results (mean $\pm$ standard deviation).....	150
A.1	Properties of deep-cycle battery types.....	163
A.2	H-Bridge modes of operation.....	167

Abbreviations

BCI	Brain computer interface
HMI	Head movement interface
SAM	Semi autonomous machine
TIM	Thought controlled intelligent machine
VFH	Vector field histogram
DWA	Dynamic window approach

Abstract

Powered wheelchairs have been dedicated to providing independent mobility to individuals who have physical or cognitive impairments and disabilities. Unfortunately, there are still a significant number of impaired people who find it difficult or impossible to use the standard powered wheelchairs due to their high level of injuries. Intelligent wheelchairs, therefore, have been developed to cater for these users, utilizing control methods and assistive navigation algorithms to help the patients safely navigate an environment.

This thesis aims at developing an intelligent wheelchair that provides functional mobility for the disabled users. Unlike a number of intelligent wheelchairs based on algorithms from the robotic field, this dissertation investigates two advanced techniques, which are Bayesian theory and artificial neural network, in building the semi-autonomous navigation system for the intelligent wheelchair. In particular, Bayesian theory is employed to determine free spaces for navigation and make adaptive shared control decisions between the users and the machine. In the meantime, the artificial neural network is recruited to learn the navigation skills through a number of patterns. During the training, the Bayesian framework is also applied to optimize the network structure as well as the weight values. After training, the network is able to navigate the wheelchair into the location of interest without colliding with surrounding obstacles.

The developed semi-autonomous navigation system was experimentally verified on the real wheelchair with a number of participants. The experimental results convincingly demonstrate that our system is able to real - time navigate in the unknown environment with safe, smooth and optimal trajectories, and that it can be a valid alternative for the people with disabilities.

Chapter 1

Introduction

1.1 Problem statement

Currently, there are a large number of people suffering from disabilities arising from a variety of causes including accidents and diseases around the world. According to United Nations (Nations 2013), 10% of the world's population, equivalent to 650 million people, live with a disability. They are the world's largest minority. In Australia, following a report (ABS 2010), there were 2.2 million Australians aged 15-64 years with disability in 2009, up from 1.7 million in 1993. The disability rate for Australians aged 15-64 years, those of 'prime working age', rose from 15% in 1993 to a peak of 17% in 2003, then returned to 15% in 2009. In 2009, just over half (51%) of the people with disability were women. In the United States, according to a comprehensive report on this population released in July 2012 by the U.S Census Bureau (W.Brault 2012), there were about 56.7 million people (19% of the population) with a disability in 2010; more than half of them reporting the disability was severe. In the United Kingdom 2011, there were over 10 million disabled adults (age 16+) equivalent to 24% of the adult population (Trust 2011).

The people with physical or cognitive disabilities and the elderly often have mobility limitations that restrict their community participation. In order to aid those populations to gain independent mobility, there are several available assistive devices such as walkers, scooters, manual wheelchairs, or power wheelchairs. The use of these devices has substantial benefits to both children and adults (Butler 1986; Nisbet 2002a; Trefler et al. 2004) in terms of intellectual as well as physical development. For instance, the power

wheelchair might assist the users to independently engage many daily activities as well as develop the physical, cognitive, communicative and social skills.

Despite the many benefits for users, standard assistive devices like powered wheelchairs have a high potential for causing negative outcomes to users with impairments. A survey (Edwards & McCluskey 2010) conducted with more than two hundred powered wheelchair and scooter users in New South Wales, Australia reported that a sizeable proportion of respondents (21%) had accidents that often resulted in personal injuries as well as damage to their devices. According to another report (Mortenson et al. 2005), cognitive deficits increase the likelihood that the older adult would have accidents (collisions) that cause injuries to themselves and bystanders, or cause damage to the surrounding physical environment.

Furthermore, not all individuals with a combination of physical and cognitive or perceptual impairments possess skills required for using powered wheelchairs. A survey (Fehr, Langbein & Skaar 2000) reported that 10 to 40 percent of participating clinicians, who desired powered mobility, would not be suitable candidates because of sensory impairments, poor motor function, or cognitive deficits that make driving safely impossible.

In an effort to address concerns relating to the usability and safety of powered wheelchairs for older adults or disabled people, researchers have proposed the concept of intelligent wheelchairs. These wheelchairs use technology to improve the functionality of normal powered wheelchairs in order to make them safer and more accessible to use.

Typically, an intelligent wheelchair consists of either a standard powered wheelchair to which a computer and a collection of sensors have been added or a mobile robot base to which a seat has been attached (Simpson 2005). The system has been designed in order to help the user to navigate in a number of different ways, such as assuring collision-free travel, aiding the performance of specific tasks (e.g., passing through doorways), and autonomously transporting the user between locations. A study (Del Castillo et al. 2006) also pointed out that nearly half of all people who find it difficult to control powered wheelchairs by the use of joysticks or other conventional methods can benefit from an automated navigation assistance of the intelligent wheelchair system. A recent report

(Simpson, LoPresti & Cooper 2008) also reaffirmed that 61% to 91% of all wheelchair users in general would benefit from the assistance of an intelligent wheelchair.

Not only rehabilitating the independent mobility for many people with disabilities, an intelligent wheelchair is also a very important training tool for children with physical disabilities (McGarry, Moir & Girdler 2012; Nisbet 1996). The research (McGarry, Moir & Girdler 2012) pointed out that the intelligent wheelchair has the ability to uncover the learning potential and facilitate the recognition of abilities in children who were previously excluded from access to independent mobility.

The benefits of an intelligent wheelchair for disabled users are very obvious, and also the demand for such systems is enormous in modern society. However, until now, designing an intelligent wheelchair system, which is capable of being commercialized, has instigated many challenges for researchers to overcome. These challenges can be broadly classed into three main areas:

- Improvements to the assistive technology mechanics
- Improvements for the user–wheelchair physical interface
- Improvements to shared control between the user and the wheelchair.

Even though numerous methods have been introduced for developing the intelligent wheelchair system to tackle the challenges in each category, very few developments have been made in order to build a system that is able to aid mobility for a great diversity of users with different levels of injuries.

1.2 Objectives of the thesis

The primary objective of this thesis is to develop a semi-autonomous control system for an intelligent wheelchair to aid mobility of elderly people and people with disabilities ranging from low to severe motor impairments. Our intelligent wheelchair system, called SAM - Semi-Autonomous Machine, is developed relying upon a laser-based navigation to provide mobile assistance. This research also focuses on cost-effective solutions that are able to

operate in unstructured indoor residential environments and can be easily retro-fitted to a range of electric wheelchair types while being minimally intrusive to the users. The task of developing the semi-autonomous navigation system is split into the several areas:

- **Instrumentation:** select and install sensors as well as necessary devices into the wheelchair in order to turn the vehicle to an ‘intelligent’ system.
- **Real-time map building:** create a map of the environment based on the onboard sensory information without prior knowledge or restructuring the environment. This enables a user to operate the wheelchair in different environment types such as the user’s home, public places or hospitals.
- **User input methods:** provide a range of input methods that facilitate controlling the system. These methods are oriented towards people with disabilities. There are both joystick-based and hands-free input methods to enable users to interface with SAM. For instance, a standard joystick might be able to accommodate many elderly users while hands-free technologies (head movement or brain computer interface) are suitable for people who find it difficult to use conventional input methods due to their either physical or cognitive limitations.
- **Free spaces:** determine spaces where the wheelchair is accessible without colliding with nearby obstacles. The information obtained from the sensors’ system is uncertain as the result of the limitation of the hardware devices; therefore, the proposed method to calculate the free spaces must be able to take the uncertainty into account in order to improve the robustness of the system.
- **Autonomous navigation:** find the most effective algorithm to safely navigate the system following the direction of travel specified by the user. The system is required to not only be able to avoid surrounding obstacles, but also to satisfy several psychological requirements for a human assistive system.
- **Semi-Autonomous mechanism:** share the control between the user and the autonomous controller. In this assistive system, the users should be obviously capable of controlling the system, and at the same time receiving the mobile support from the autonomous controller as they lack fine skills to navigate the system by

themselves. The proposed shared control strategy should help the system achieve that goal in the real-time.

1.3 Thesis contributions

The most important contribution of this thesis is the development of a semi-autonomous navigation system for an intelligent wheelchair. In addition, throughout building the system that is a test platform for the robot control architecture and sensor research, particularly machine vision, this study also contributes to human-robot interaction, adaptive shared control architectures, and novel input methods for the human-machine. In particular, the research is demonstrably capable of performing all the objectives as mentioned in the section 1.2.

- With add-on style design, the prototype developed by this research might have a straightforward application to a number of commercially powered wheelchairs in order to transform them into an intelligent system. In another direction, the additional devices can be easily removed from the system to return to an originally standard platform.
- The real-time map is built on the basis of combining data from different sensors. Uniting onboard sensory information helps to increase the reliability of perception about the environment of the wheelchair. Moreover, the advantage of the proposed mapping technique is to not only simplify the computation, but also limit a number of approximation steps that typically cause errors in many critical situations, thus improve the system's real-time capabilities.
- The proposed technique to determine the free space in this research takes the uncertainty rising from the measure errors and the capability of passing an opening of the autonomous navigation algorithm into account. As a result, the technique would improve the robustness because it lessens the effect of measuring errors and autonomous algorithm errors. Many other autonomous systems in the robotic field would benefit from this method in calculating the free spaces.

- The autonomous navigation algorithm is implemented based on a Bayesian neural network. Instead of using complicated discrete formulas to describe the control rule for the wheelchair, in this thesis, the system is guided to learn how to react in certain navigation situations. The greatest advantage of this approach is that it creates the obstacle avoidance controller which safely navigates, with human driving skills, through a pattern learning process rather than rigorous responses resulting from a route program paradigm. The technique created by this research is also useful and feasible for many mobile applications in the future.
- Unlike many other intelligent wheelchairs that restrict interaction between the user and the system, this thesis proposes an adaptive control architecture in which the user is free to control the system the way he/she likes, and the machine always cooperates with the user to make sure the system is safe. The contribution is to build an effective shared control framework relying on the Bayesian theory to real-time adapt the changes of the user's inputs and the environment. This work is not only valid for this project, but also very useful in many human – machine interaction applications.

Another major contribution of this thesis is to provide an up-to-date literature review into relevant areas. The reviews in chapter 2 provide a broad but compact overview into intelligent wheelchair systems. The literature review helps to ascertain current trends towards intelligent wheelchair systems, thus providing direction for the research carried out in this thesis.

1.4 The structure of the thesis

The goal of this thesis focuses on developing a semi-autonomous navigation system for the intelligent wheelchair in order to assist people with a variety of impairments. The work consists of six chapters in which this first chapter is for the introduction and the rest is structured as follows:

Chapter 2 – provides a background for intelligent wheelchair systems. Then, a review of the various platforms around the world is introduced. Along with the various platforms, their core components, i.e., sensors, user input methods, and navigation assistance approaches are also exposed. At the end of the chapter, a discussion about the advantages and disadvantages of the current intelligent wheelchair systems takes place to find a gap for this research.

Chapter 3 – details our intelligent wheelchair design that is known as the “add-on” design style. In chapter 3, we also introduce our method in order to build a real-time map for the wheelchair combining sensory information from different onboard sensors. From the obtained real-time map, a number of free spaces are identified by the Bayesian recursive technique. The uncertainty rising in the measurement errors as well as the capability of passing an opening of the autonomous algorithm is modelled into the probabilistic information, and then the Bayesian rule is recursively applied to find the free spaces. Finally, two experiments are reported at the end of the chapter to demonstrate the effectiveness of the proposed method.

Chapter 4 – describes an advanced autonomous navigation algorithm that is capable of operating in unknown environments. In order to work effectively in such a real-time application, the feed-forward neural network architecture, capable of approximating any continuous functional mapping, is introduced to our wheelchair. Then, the Bayesian regularization in combination with Levenberg – Marquardt training is developed to apply to this practical application. Under this framework, we are able to estimate the most optimal network structure as well as increase the generalization of the feed forward neural network on the basis of this application’s training data. This chapter also describes a number of experiments designed to collect data for the purpose of training neural network. Finally, the results of the experiments about the network training and real-time implementations in order to evaluate the trained network are presented.

Chapter 5 –is about the issue of sharing the control system between the user and the machine. The semi-autonomous navigation system is built to accommodate

varying degrees of autonomy depending on the driver's skills and type of input methods that the impaired user is able to employ. After the introduction section, an advanced Bayesian shared control strategy architecture and its practical implementation for our intelligent wheelchair are presented which demonstrate how to cooperate the human user inputs and environment information to make adaptive shared decisions. Finally, two experiments will be presented. The first experiment is to demonstrate that the system using the Bayesian shared control framework is able to make the most appropriate shared control actions to navigate in the real-time. The second experiment implemented with a group of able - body participants is to evaluate the performance of the semi - autonomous system on the basis of quantitative and qualitative measurements.

Chapter 6 – provides an overall discussion of the performance of the developed methods in the thesis. Also some future works to improve this project will be suggested.

At the end of the thesis, some appendices and the bibliography relating to the previous chapters are included for the reader's reference.

1.5 Publications related to the thesis

There are four international conference papers of the Institute of Electrical and Electronics Engineers (IEEE) related to this thesis. They are listed as follows:

- **A.V. Nguyen**, L.B. Nguyen, S.S. Wang & H.T. Nguyen, (2011), 'Development of a Bayesian Recursive Algorithm to Find Free-Spaces for an Intelligent Wheelchair.', *33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Boston, Massachusetts USA, 2011, pp. 7250-3.
- **A.V. Nguyen**, L.B. Nguyen, S.S. Wang & H.T. Nguyen, (2012), 'Development of a Bayesian Neural Networks to Perform Obstacle Avoidance for an Intelligent

Wheelchair', *34th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, San Diego, California, USA, 2012, pp. 1884-7.

- **A.V. Nguyen**, L.B. Nguyen, S.S. Wang & H.T. Nguyen, (2013), 'The Advancement of an Obstacle Avoidance Bayesian Neural Networks for an Intelligent Wheelchair', *35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Osaka, Japan, 2013, pp. 3642-5.
- **A.V. Nguyen**, L.B. Nguyen, S.S. Wang & H.T. Nguyen, (2013), 'Shared Control Strategies for Human - Machine Interface in an Intelligent wheelchair', *35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Osaka, Japan, 2013, pp. 3638 - 41.

Chapter 2

Literature Review

2.1 Introduction

Commercial electric-powered wheelchairs traced to the 1950s (Woods & Watson 2003) have been providing functional mobility for people with both lower and upper extremity impairments. With the assistance of the wheelchair, these people are capable of moving around their home, going to work or doing their daily work independently. However, challenges of safely and independently operating a wheelchair can result from various overlapping motor, perceptual, or cognitive impairments such as spinal cord injury, or cerebral palsy.

Along with developments in the robotic fields, a large number of intelligent wheelchair platforms have emerged to accommodate the needs of individuals with disabilities who find it difficult or impossible to operate a conventional power wheelchair, such as one with a joystick. Generally speaking, an intelligent wheelchair, also called a smart wheelchair, is any motorized platform with a chair which is equipped with sensors and computing devices, such as a computer, in order to provide a certain level of autonomous mobility to assist the user. It is important that the system is designed to communicate in various ways with the user plus provide mobility.

This chapter will be organized as follows. At first, we briefly introduce some core components that construct an intelligent wheelchair system and the necessary requirements

for the such system. Then the research – based intelligent wheelchair systems and their navigation algorithms are covered. Note that we only focus our review on the types of platforms that are most related to our research. Finally, the challenges and shortcomings of current intelligent wheelchair systems are discussed at the end of the chapter.

2.2 Intelligent Wheelchairs

2.2.1 The system

In a standard powered wheelchair, a user provides his commands in terms of electrical signals via traditional interfaces such as a joystick, and these signals are directly transferred to a controller to manoeuvre the wheel motors. In that way, the wheelchair moves to the user's desired directions. However, as some users are unable to give proper commands due to physical or cognitive limitations, this might cause the wheelchair to collide with surrounding obstacles resulting in injuries for the user (Edwards & McCluskey 2010). An intelligent wheelchair, therefore, is designed to prevent such collisions from happening.

Despite many platforms which are designed for a variety of user types, an intelligent wheelchair strictly speaking comprises either a standard power wheelchair to which a computer and a number of sensors have been attached or a mobile robot base to which a seat has been added (Simpson 2005). Unlike a robot which navigates by itself, an intelligent wheelchair typically is known as a semi-autonomous system whereby both a human user and the machine share the control of the system depending on strategies predetermined by the designer. General block functions of an intelligent wheelchair system can be illustrated as the figure 2.1. They consist of user interfaces, sensors, semi-autonomous control, and wheelchair motor controllers.

Rather than directly employing user's commands to control the wheelchair motors as a standard platform, the user's input signals are fed into the semi-autonomous control block which decides particular control signals for these motors based on the relationship between the user's intention and the surrounding environment. Because of this advanced control principle, the user is able to utilize many different kinds of interface to give control signals.

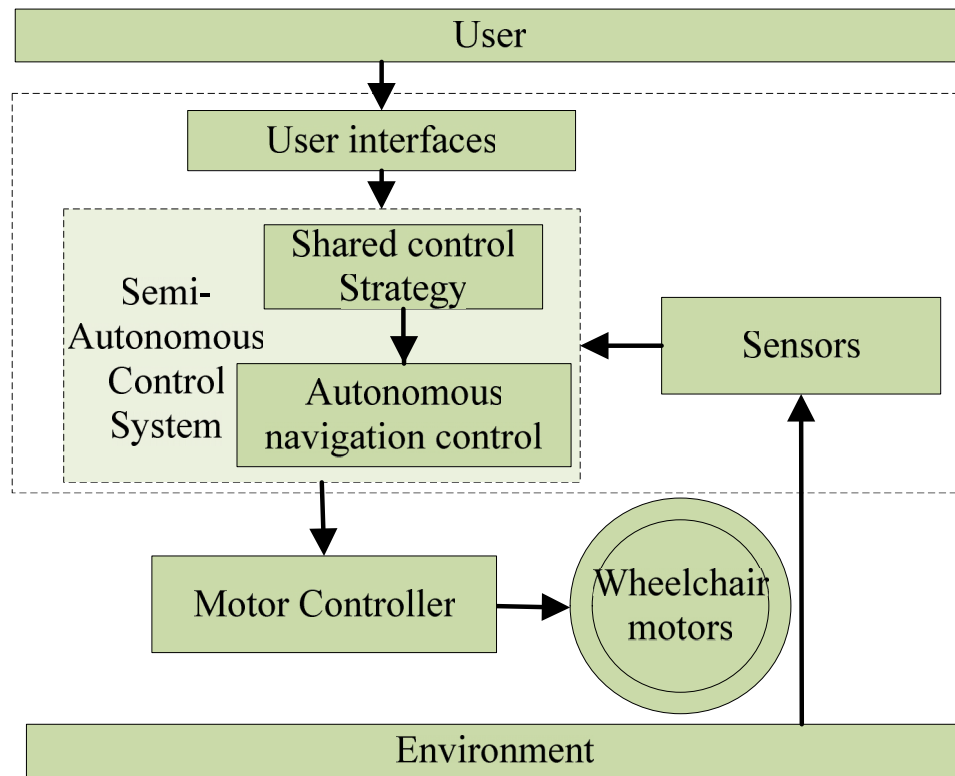


Figure 2.1: The block functions of an intelligent wheelchair

This plays a significant role in enabling major users with different levels of injuries to control the wheelchair. A variety of user input methods has been invented and each specific type of input method is typically suitable for the particular injury or disability of the user. Besides traditional input methods like joysticks suitable to those who are capable of using upper limbs, hand-free methods such as head movement, or Brain control interface are indispensable to those with more severe disabilities who fail to move their hands.

One of the most important parts in the intelligent wheelchair system is sensors. The system is always equipped with a number of sensors perceiving the wheelchair's surrounding environment information. There are many requirements for sensors that are difficult to satisfy for such a system. For example, they need to be accurate, small, lightweight, inexpensive, consume little power, and be robust to stand up to environmental conditions. In practice, each sensor type has its own set of problems: ultrasonic range finders fail on sound absorbent surfaces and IR range finders can be fooled by light absorbent surfaces. Cameras require sophisticated algorithms to extract the desired information as well as filter

out the unwanted details and noise, and so on. To tackle practical restrictions, an intelligent wheelchair system typically fuses environment information from multiple sensors.

The central part in the system is a semi-autonomous control part that is comprised of shared control strategy and an autonomous navigation control. The purpose of autonomous navigation control is to produce control signals in considering obstacles surrounding the wheelchair and the desired direction of travel. It does so by gathering environment information through sensors. Data from the sensors once processed and the travel direction are considered as inputs of a designed autonomous algorithm whose outputs are control signals. The designed autonomous navigation method generally plays a decisive role for the system's safety as well as the comfort for the user.

Characteristically, control signals produced by the user and the autonomous navigation method are not the same while both signals intend to control the wheelchair. Therefore, a shared control strategy is needed in order to integrate the human inputs with autonomous behaviours of the system. The strategy should be established by weighing up many factors. Although the autonomous navigation part might guarantee to navigate the system collision-free, the user is always reluctant to use a system that is always led by the machine. Therefore, a good shared control strategy should admit the user to be involved in the navigation process as much as he/ she can, while making certain the system safety is always assured.

2.2.2 Requirements of the Intelligent Wheelchair Application

It is natural to think the intelligent wheelchair system to be a human – machine system in which both the subjects have to work together to achieve the common tasks. There are, thus, various constraints specific to the application of mobility assistance for people with disabilities (Bourhis & Agostini 1998; Kruse et al. 2013; Mazo 2001). The following specificities are considered as the most essential things for designing an intelligent wheelchair system.

Versatile system: An intelligent wheelchair system should be designed with the aim of being versatile. This is on the basis that the designed prototype intends working with a great diversity of users and situations. The environment might be the user's home, or public structure like a rehabilitation centre or a hospital. Possible paths as well as environmental types are therefore very variable. In addition, the wheelchair potential users also have a variety of cognitive and physical attributes.

Psychological acceptability: As the wheelchair operators are situated on the mobile base, psychological acceptability problems are imposed on the system. For example, the system must avoid sudden speed or direction variations during automatic or manual motions. More importantly, it is necessary that the users feel that they are in control of the chair at all times. The prototype should guarantee that the driver can react if any problems arise that may represent a risk despite the degree of autonomy. Overall, it is necessary that the chair inspires confidence in the user.

Safety and Robustness: These aspects are especially sensitive in such an application where the pilot is located on the machine, and furthermore has limited cognitive and physical capacities. The system is therefore designed to be able to detect obstacles, holes, and many other dangerous situations. The system's robustness is reflected in that it can exclude all blocking situations. For example, as the wheelchair may be blocked due to some safety criteria, the prototype should allow the pilot to be able to resume control at any time in the manual mode.

Low cost system: It is essential to take into account the implementation cost price in the application of high technology technical aids for people with disabilities. It is therefore necessary to use a commercialized powered wheelchair as a mobile base. The choice of sensors also strongly depends on the cost criterion.

In summary, there are various specificities for designing the intelligent wheelchair system, especially robustness, psychological acceptability, versatility, and low cost performance compromise. In the designing system, we find that the combination of these requirements always conflicts, the satisfaction for all these criteria is impossible, but an acceptable

compromise between these requirements can be achieved through selecting appropriate control architecture for the system.

In the next section, we will review a number of intelligent wheelchair systems that have been building around the world in an effort to accommodate the disable user's demands.

2.3 Intelligent Wheelchair System Review

An intelligent wheelchair system, University of Michigan, US (1993 – 2002), known as Navchair, is equipped with 12 ultrasonic sensors and an onboard computer as in the figure 2.2. A user can employ either joystick (Simpson & Levine 1999) or voice control (Simpson & Levine 1997) to operate the system. In order to prevent the wheelchair from colliding with obstacles, a number of obstacle avoidance tasks relying on vector field histogram (VFH) are designed for particular environment types such as door passing task, wall following task, and so on. Then, depending on the user's intention and surrounding environment, the system automatically activates the corresponding task to drive the wheelchair.



Figure 2.2: Intelligent wheelchair system, University of Michigan (IEEE, 1999)



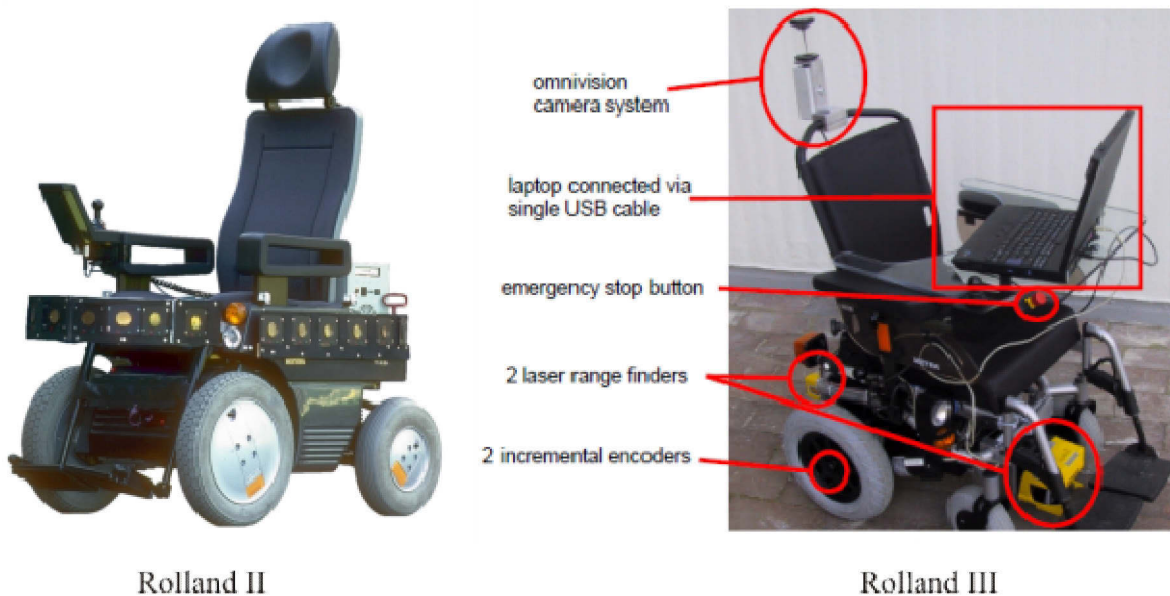
Figure 2.3: Intelligent wheelchair system, Osaka University (IEEE, 2003)

The intelligent Wheelchair System from Osaka University Japan (1998-2003) concentrates on the human-interface issue and implements conventional autonomous capabilities. The system has a pair of stereo CCD cameras to capture the facial image of the user, and vision, ultrasonic, and other sensors to recognize the surrounding environment as in the figure 2.3. The head pose and gaze direction are detected by two stereo CCD cameras considered as a user input method (Matsumoto, Ogasawara & Zelinsky 2000), one facing toward the user, second facing forward. The user gives a command to the system with head gestures, interpreted by an inward-facing camera. An outward-facing camera tracks targets and allows the user to control the wheelchair with gestures when out of the wheelchair. The gaze detection is then combined with the current position of the wheelchair, determined via the environment observing sensors and an internal map, in order to estimate a user's intention (Adachi et al. 2003; Adachi et al. 2004).

To reduce the user's burden of operation, and safely navigate, this system developed some autonomous tasks, like target tracking, follow walls, and avoid obstacles based on the outward-face camera and other observing sensors (Kuno, Shimada & Shirai 2003). After determining the user's intentions from observing the face, the system selects an appropriate autonomous navigation function to operate the vehicle to the target. In particular, as the user looks straight ahead for a short time, the outward-facing camera identifies the target and moves toward it. The outward-facing camera is also used to identify pedestrians,

determine where the user is looking, and move the chair in the opposite direction to avoid collisions.

The intelligent wheelchair system from University of Bremen, Germany (1997- 2005) is called Rolland. There were three different prototypes throughout the whole development period. The first prototype developed by (J. Kollmann 1997) is based on sonar, IR and bump sensors to avoid collisions. The system is able to autonomously navigate between points on its map as well as some operating modes like wall following or door passing (Röfer 1998). The Rolland II (A. Lankenau 2000, 2001, 2002, 2003), as the figure 2.4, only recruits sonar sensors but has a more complicated autonomous navigation algorithm. Three operating modes (turn in place, wall following and trajectory playback) are provided and automatic mode transitions triggered by obstacle density. The last prototype (C. Mandel 2005) is very different compared to its two previous designs in both hardware and software terms. It used two laser range finders, encoders and Omni-vision camera system for solving fundamental tasks like self-localisation, mapping and autonomous navigation.



Rolland II

Rolland III

Figure 2.4: Intelligent Wheelchair System, University of Bremen

The intelligent wheelchair system from University of Pennsylvania (2002- 2007) is equipped with an onboard processing computer and a suite of sensors as shown in the figure 2.5. The Omnidirectional camera provides a 360 degree view around the wheelchair,

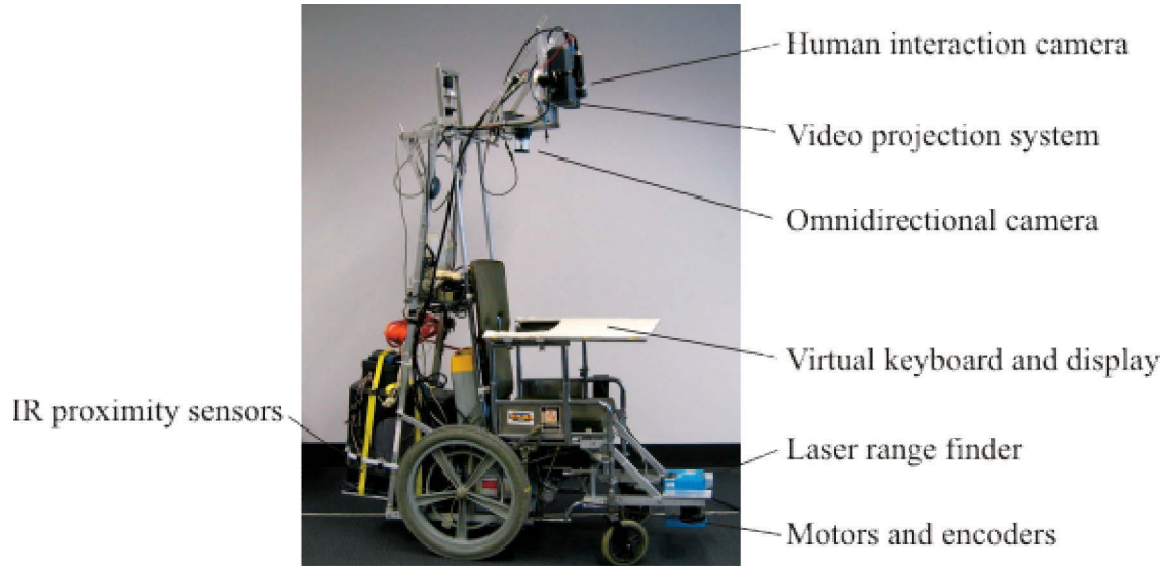


Figure 2.5: Intelligent wheelchair system, University of Pennsylvania (IEEE 2007)

the image is projected onto the lap tray so that a user can view and send commands via a visual interface. A laser range finder measures obstacles in front of the wheelchair while IR proximity sensors detect obstacles on the back. Lastly, encoders provide a dead reckoning system (Parikh et al. 2007).

The user initially selects a desired destination, the system automatically plans a path; will choose modes to take the user there. There are several modes of operations including approaching and passing designated doors, travelling to a specific target, hallway navigation, and collision avoidance (Parikh et al. 2003).

The intelligent wheelchair system from University of Zaragoza (2008-2010) is designed based on two onboard computers, a SICK LMS200 planar laser, and a screen. The two computers are connected via RS-232 and Ethernet: One is to perform a low level control which real-time controls the differential-drive rear wheels, and the other is responsible for the medium-level control performing the navigation computations. A user operates the system via selecting a target, also called an intermediate goal, in the visual display that is a 3-D online environment visualization built from sensor measurements. As shown in the figure 2.6, the goal selection is implemented through either touching on the screen (Montesano et al. 2010) or using a P300 neurophysiologic protocol (Iturrate et al. 2009).

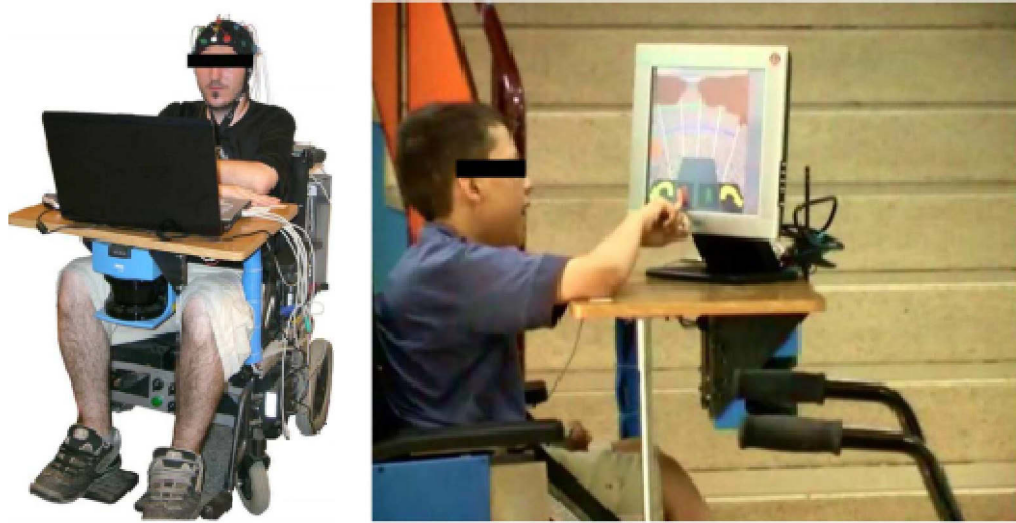


Figure 2.6: Intelligent wheelchair system, University of Zaragoza (IEEE 2009, 2010)

When a destination is chosen, the autonomous navigation system drives the wheelchair to the target while avoiding obstacles measured by the laser sensor. Two modules, a model builder and a local planner, are built and integrated to provide several autonomous functionalities. The model builder aims to model the static obstacles, free space, updating the map, detecting and tracking the moving obstacles while the local planner computes the local motion based on the combination of tactical planning (Ranganathan & Koenig 2003) and reactive collision avoidance (Minguez & Montano 2004).

The intelligent wheelchair system from University of Toronto, Canada (2007-present) is built on a single 3-D time-flight infrared laser range sensor in the front of the wheelchair, a

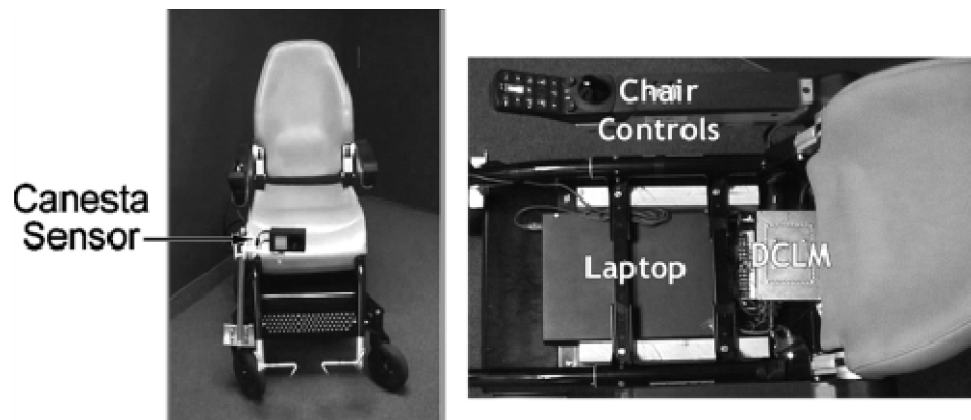


Figure 2.7: Intelligent wheelchair system, University of Toronto (IEEE 2007)

laptop computer, and a direction control logic module (DCLM) (Mihailidis et al. 2007; Viswanathan et al. 2011) as the figure 2.7. A depth image produced by the 3-D sensor is used to construct an occupancy grip map that represents the distance of obstacles to the wheelchair in 2-D-top-view. Based on the occupancy grip and a user's command via a joystick, DCLM decides the motion of the wheelchair. This system also recruits a path planner to suggest a possible action for the user through an audio prompt.

The intelligent wheelchair system from University of Leuven, Belgium (2002-2010) – also called Sharioto – is equipped with a joystick or joystick chin. The wheelchair in the figure 2.8 is attached with 9 infrared sensors, 16 ultrasound sensors, and 4 ultrasonic sensors to measure surrounding obstacles, 2 potentiometers for measuring the orientation of the front castor wheels, 1 lidar (light detection and ranging sensor), and a gyroscope for rotational velocity estimation. Although a VFH based collision avoidance algorithm is provided, the main work focuses on the use intention estimation (Huentemann et al. 2007; Nuttin et al. 2002; Vanhooydonck et al. 2010). The user's signals from joystick are combined with surrounding environment information to find the 'real' user's intention using Bayesian framework in (Huentemann et al. 2007) and artificial neural network in (Vanhooydonck et al. 2010).

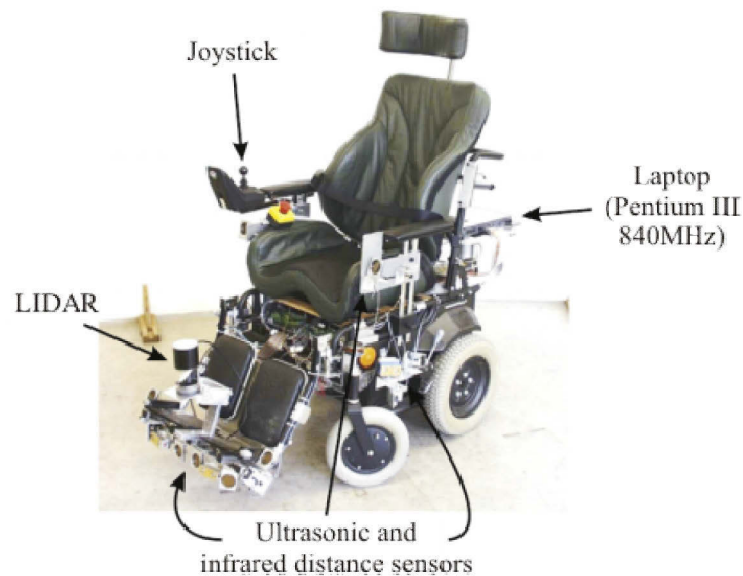


Figure 2.8: Intelligent wheelchair system, University of Leuven (ELSEVIER 2010)

The intelligent wheelchair system from University of Pittsburgh, US (2002-Present) is designed based on add-on components, called the drive-safe system (DSS), attached on a powered wheelchair as the figure 2.9. These components have been developed throughout a number of works (LoPresti et al. 2011; Simpson et al. 2004; Simpson, Poirot & Baxter 2002). The DSS has five sensor modules providing coverage on all sides of the wheelchair and joystick translator module to communicate with the wheelchair controller. Each sensor module comprises of five ultrasonic rangefinders, five IR rangefinders, two bumper inputs, one speaker, and three status light-emitting diodes. This system allows a user to control the wheelchair by joystick, and different assistant navigation modes like door-crossing, wall – following, and corridor crossing are also provide when needed.



Figure 2.9: Intelligent wheelchair system, University of Pittsburgh (JRRD 2011)

Based on the relationship between the user's continuous signals (including a speed and a steering angle) and surrounding obstacle's position, the system decides whether to directly use these signals or activate assistant modes to control the wheelchair. By doing so, it both accepts the maximum user's involvement and protects the system from situations that might be subject to collisions.

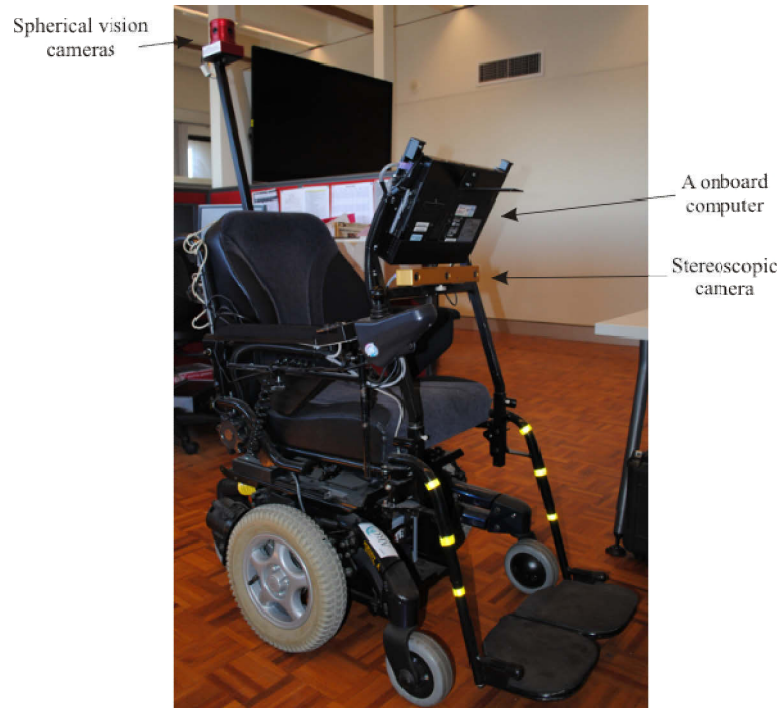


Figure 2.10: Intelligent wheelchair system (TIM), University of Technology, Sydney

The intelligent wheelchair system from University of Technology, Sydney (UTS), Australia (2007-Present), called TIM (Thought-controlled Intelligent Machine) as the figure 2.10, is designed by attaching some additional components onto a commercial powered wheelchair, including stereoscopic cameras, spherical vision cameras, an onboard computer and wheel encoders. To provide obstacle avoidance tasks for the system, the combination of stereoscopic cameras and spherical vision cameras is implemented to detect surrounding obstacles (Nguyen, Su & Nguyen 2010; Nguyen et al. 2012). Then, based on the obstacle data, a real time obstacle avoidance algorithm relying on the Vector Polar Histogram assists the system to navigate through unknown environments without collisions. A number of different methods are available to control the system such as Brain computer interface (BCI), head movement interface.

The intelligent wheelchair system from University of Saitama, Japan (2011 - present), shown in the figure 2.11, is equipped with four types of sensors: a standard webcam, a Camera (Kinect), a laser range finder, and an inertial measurement unit sensor. The webcam is mounted in front of the user and plays a role as a user interface on the basis of

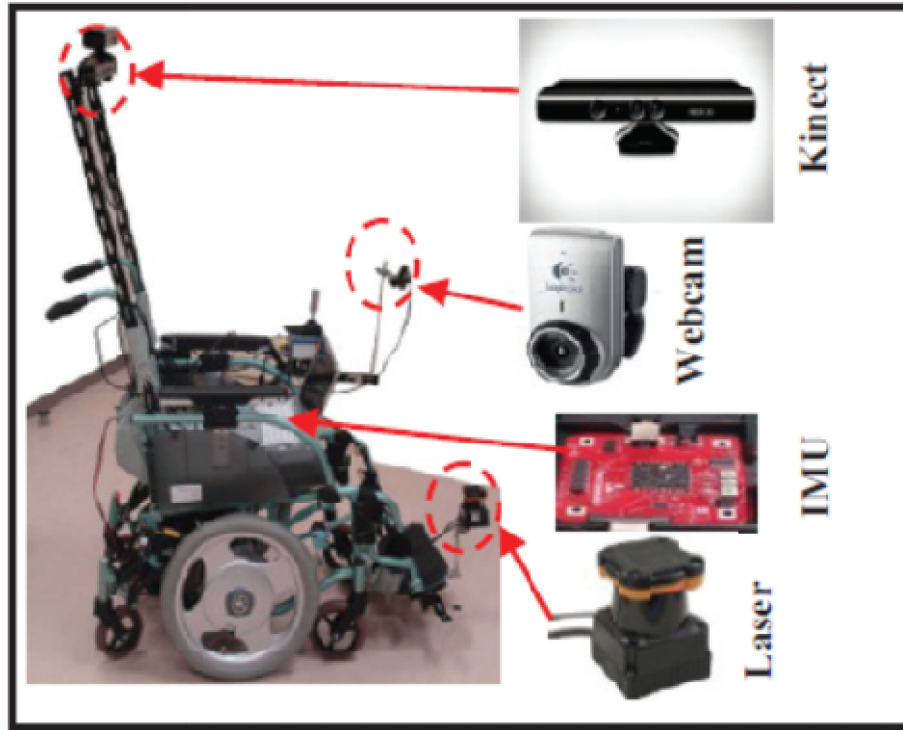


Figure 2.11: Intelligent wheelchair system, University of Saitama (ELSEVIER 2012)

gazing detection in which the value of the yaw angle of the user's face is utilised to classify one of three pre-defined commands: move forward, turn left and turn right. In terms of obstacle avoidance, this system builds a safe area called a safety map by combining the laser and the camera information (Bebis et al. 2011; Tomari, Kobayashi & Kuno 2012), and a VHF strategy with a slight modification is applied to assist the system to avoid obstacles. This wheelchair can work with either manual control mode or semi-autonomous control mode (Tomari, Kobayashi & Kuno 2012). The switching between working modes is manually carried out via a switch depending on how long the user leaves it on. The switch is also used in order to stop the system.

2.4 Autonomous Navigation Algorithms

This section provides an overview of several reactive collision avoidance algorithms that only consider the immediate environment surrounding the wheelchair. Such methods

generally have low computational complexity and are thus appropriate for real-time operation in dynamic environments. Although a variety of obstacle avoidance methods for intelligent wheelchairs have been reported, they can be briefly divided into two types of paradigm which are known as route program paradigm and learning machine paradigm.

2.4.1 Route Program Paradigm

As its name implies, route programs are constituted by step by step code for instructing the wheelchair to follow. In other words, the principle of this paradigm to control the machine is to model the surrounding environment by specific and approximate formulas or code statements. The following is a brief of the most popular methods that have been applied in the intelligent wheelchair field.

One of the most popular obstacle avoidance methods of this type is the potential field navigation method, as originally described by (Khatib 1985) and recently refined for the wheelchair (Petry et al. 2010). This method employs the philosophy of attraction and repulsion of a charged particle in the potential field. It models the vehicle as a charged particle based on (Lozano-Perez 1983) while obstacles provide repulsive forces and a target is represented as an attractive force. A resultant force from the field upon the vehicle is then used to provide suitable navigation commands. The most interesting feature of potential fields is conceptually simple, and straightforward to implement, while the main problems with this method are that it exhibits oscillatory behaviour in narrow passages and struggles to pass through closely spaced obstacles as the vehicle often reverses from such situations.

Another real-time obstacle avoidance method is vector field histogram (VFH) which was first proposed by (Borenstein & Koren 1991), and widely used for the wheelchairs (Kurozumi et al. 2010; Levine et al. 1999; Lopes, Nunes & Vaz 2010; Tang et al. 2012). The VFH method typically recruits range-bearing measurement sensors to build a 2-D histogram around the vehicle which contains both new sensor readings and a history of previous readings. The 2-D histogram is then transformed into a 1-D polar histogram,

where the magnitude of each sector represents a measure of obstacle density in that direction. The data reduction is next performed by thresholding a smoothed version of the polar histogram. Finally, the resulting steering angle is determined by finding a suitable valley in the polar histogram with which the vehicle can be aligned. The benefits of VFH are relatively robust, low computational requirements. However, the VFH method requires the manual finetuning of the threshold value which decides whether a sector in the polar histogram is a peak or a valley.

Unlike VFH that use distances to obstacles as its main input, the dynamic window approach (DWA) operates directly in the velocity space. This method is introduced by (Fox, Burgard & Thrun 1997) and is often used in the intelligent wheelchair system (Demeester et al. 2005; Yokozuka et al. 2012). The DWA consists of two key steps: to find the set of possible (both translational and rotational) velocities that satisfy some criteria, and maximize an objective function. In the first step, the method reduces the search space for possible velocities with the following three criteria:

- Circular trajectories: only circular trajectories described by pairs of translational and rotational velocities are considered.
- Admissible velocities: only safe pairs of velocities are considered, i.e., velocities which allow the vehicle to stop before it reaches the closest obstacle on a trajectory.
- Dynamic window: restricts the velocity pairs that are possible to achieve within a short time interval, given the acceleration limits of the vehicle.

In the next step, the greatest velocity pair is selected from velocity pair space that has been reduced by the above criteria by finding which one maximizes the objective function. The objective function is a weighted sum of three components: heading to the target, clearance between the closest obstacle, and forward velocity of the vehicle.

The nearness diagram (Mandel et al. 2009; Minguez & Montano 2000) is composed of analysing a situation from two polar diagrams, which navigates based on gaps in the environment. These diagrams are constructed from both current and previous range-bearing sensor readings. One diagram, representing the nearness of the obstacles to the centre point of the vehicle, is used to find discontinuities and valleys in the environment, a similar idea

to the peaks and valleys with VFH. The other diagram, representing the nearness of the obstacles to the vehicle boundary, identifies the safety level between the vehicle and the nearest obstacle. The information extracted from these diagrams is applied to classify the situation as one of five general situations which covers the possibilities among the vehicle location, goal location, and obstacle configuration. From this, suitable translational and rotational velocity commands are obtained.

In conclusion, it is obvious that the approaches belonging to the route program paradigm are easily understood and implemented because they are explicit and rigorous. Nevertheless, it is sometimes very difficult to describe the desired, exact vehicle behaviour in many practical situations by proper formulas or code statements. To tackle the difficulty, some systems (Galindo, Gonzalez & Fernandez-Madrigal 2006a; Simpson et al. 2004) are typically implemented with a number of different algorithms associated with different tasks such as general obstacle avoidance task, wall following task, door passing task, and so on, then activate a particular task as the conditions of the task are met. By doing so, although it can provide some more flexibility for the system in terms of adaptation with environment, the complexity of the system significantly increases.

2.4.2 Learning Machine Paradigm

With the advance of computational technology, the learning machine paradigm has been emerging in recent years. This paradigm replaces explicit programming by training the machine to navigate through a cluttered environment while concurrently avoiding surrounding obstacles. Suitable experimental designs and problem formulations which instruct tasks for the wheelchair can be demonstrated by a human operator. Subsequently, the demonstrated data can be abstracted and learned by the vehicle, by constructing computational models with existing machine/statistical learning techniques such as neural networks and support vector machines (Mjolsness & DeCoste 2001; W.T.Miller, R.S.Sutton & P.T.Werbos 1990).

In (M.Nechyba 1998), outdoor on-road driving systems with learning by demonstration paradigm have been investigated. The human driving strategies have been abstracted and transferred into an autonomous driving system. The (Matsumoto et al. 2000) proposed a model where a mobile robot learnt routes in surroundings, like corridors, by simply storing a sequence of images obtained during human-guided, movement directed autonomous movement. The (Kidono, Miura & Shirai 2002) just used single human demonstration whereby an operator guided a mobile robot to a destination by remote control while the robot concurrently built a map by observing the surrounding environment using stereovision. Then, the robot could localize and find the shortest path to the predetermined destinations.

In the intelligent wheelchair navigation learning area, a group from the Chinese University of Hong Kong (Chow & Yangsheng 2006; Hon Nin, Yangsheng & Tso 2002) has applied the paradigm of learning for an intelligent wheelchair. They implemented a number of demonstrations on the same obstacle course in which sonar sensor readings were recorded as inputs and human demonstrated commands corresponding to those inputs were known as outputs. These patterns were used for either learning in cascade neural network with node-decoupled extended Kalman filtering (Hon Nin, Yangsheng & Tso 2002) or building an explicit lookup table (Chow & Yangsheng 2006). After the learning process, the wheelchair has the capacity to complete autonomous navigation tasks in the demonstrated route. Similarly, (Hamagami & Hirata 2004) also uses a neural network for learning an obstacle avoidance action. Analogously, sensor outputs were treated as inputs of the network and four discrete network outputs are forward, left turn, right turn, and backward. A number of weight values of the network is determined by using an unsupervised genetic programming - based learning technique.

Unlike above methods that only applied in simulations or static environment, (Trieu, Nguyen & Willey 2007, 2008a) introduced a Bayesian neural network obstacle avoidance that enables the wheelchair to successfully operate in unknown environments. In this framework, the wheelchair learns human skills in navigation via various patterns collected from a number of different environment types. The training process for the feed-forward neural network has taken place off-line under the supervision of Bayesian rules to increase

the generalization. After training, the wheelchair is not only able to avoid obstacles in unknown environments, but also provides trajectories resembling those provided by an experienced driver.

2.5 Shared Control Strategies for Intelligent Wheelchairs

Unlike the robot control architecture, the intelligent wheelchair always has two control sources: the user and the autonomous navigation controller. There is an important characteristic associated with such an application of mobile robotics to the assistance of people with disabilities: on one hand, the user having reduced physical capacities seeks the mobile assistance that hence has to be natural to the user; on the other hand, the human, located on the mobile base, has a direct visual return of the wheelchair's environment. Based on this feature, it is necessary to attain the cooperation between the man and the wheelchair in the achievement of the mobility task. In so doing, the system not only satisfies the psychological comfort criterion but also improves safety with low cost. In particular, the person might bring his ability to perceive and interpret the close environment, and his global comprehension of the situation. The machine brings its autonomous mobility skills of course. In the literature, many approaches that aim to integrate the human inputs and autonomous behaviours have been reported and they vary from fully autonomous to full human control.

In the fully autonomous mode, the intelligent wheelchair operates in a manner very similar to autonomous robots (Carlson & Demiris 2012; Galindo, Gonzalez & Fernandez-Madrigal 2006b; Hemachandra et al. 2011; Seki et al. 2000). The user only selects a destination on the map, and the wheelchair then automatically plans and executes a path to reach the chosen target. In order to achieve the target, the system usually requires a complete map of the operating environment. It is clear that in the fully autonomous mode, the user is almost uninvolved in the navigating process except when selecting the target. Thus, this mode type might be suitable for those who have insufficient ability to plan and execute paths to a target and spend most of their time in a particular environment. Nonetheless, many other users might be psychologically reluctant to use the system which is led by the machine.

On the contrary, the fully human control mode requires a user to take charge for the majority of planning and navigating duties while the wheelchair only provides its assistance to stay away from obstacles (Carlson & Demiris 2012; LoPresti et al. 2011; Parikh et al. 2007; Tomari, Kobayashi & Kuno 2012). Although these systems do not need prior knowledge of the operating area, they do need users to provide continuous throughput via joystick as in the case of manual control, and thus this mode is appropriate for those who can effectively plan and execute a path to a destination.

The final mode of shared control in intelligent wheelchair systems is the semi-autonomous mode that is able to accommodate a wider range of needs and abilities of users. The semi-autonomous mode offers both autonomous navigation and the human user to engage the navigation spontaneously to perform a task. In this mode, the user lets the system begin to behave autonomously; however, he/she can interrupt or change the system's behaviour at any time the task is executing. The advantage of the semi-autonomous mode is the control flexibility for the system that not only satisfies the psychological desire of the driver but also improves the safety of the system. Nevertheless, the difficulty of this mode is to select a suitable mechanism in order to integrate appropriately both the autonomous commands and the explicit intentions of the users, especially those with disabilities.

There are semi-autonomous methods that are based on the user's ability to decide the level of combination between the user and autonomous commands. Some simply estimate the consistency of the user inputs (Parikh et al. 2007; Urdiales et al. 2011). Others measure the user's driving skills or their health conditions, and then decide at which level the autonomous system needs to assist users (Urdiales et al. 2010; Vanhooydonck et al. 2010). These approaches are usually applicable for only a specific individual, and have difficulties in application for a wide range of people.

Other shared control methods combining autonomous and user's commands are based on the consideration of both environment information and the user's commands concurrently. For example, some methods use the information from the environment and user commands to activate the appropriate available autonomous tasks (Argyros et al. 2002; Simpson & Levine 1999; Trieu, Willey & Nguyen 2009). Others can rely on decision making rules for

the assigned degree of user control (Carlson & Demirir 2012; Huentemann et al. 2007; Lopes, Nunes & Vaz 2010; Trieu, Nguyen & Willey 2008b).

2.6 Discussion

Up to this point, we clearly see that there are many innovative and promising intelligent wheelchairs that have been developed in order to assist cognitively impaired individuals. In spite of the fact that these wheelchair systems employ a variety of controls and sensors to assist users, only very few standardized design methods of intelligent wheelchair have been introduced with the aim of commercializing their design at the moment (e.g. (LoPresti et al. 2011; Simpson et al. 2004)). As clearly stated in the first chapter of this thesis, the purpose of this research is to design an intelligent wheelchair system that is able to become a commercialized product to meet the demands of a wide range of disabled people. At our Centre for Health Technologies, University of Technology, Sydney, a standardized prototype for intelligent wheelchairs is being developed which can be easily retrofitted to a range of electric wheelchair types with minimal modifications to the underlying wheelchair (Nguyen et al. 2011; Nguyen, Nguyen, et al. 2012; Nguyen et al. 2013a, 2013b). This work continuously devotes its attention to further investigate and expand that proposed system (Trieu, Nguyen & Willey 2007, 2008a, 2008b; Trieu, Willey & Nguyen 2009).

Like many other recent projects (Montesano et al. 2010; Tomari, Kobayashi & Kuno 2012; Vanhooydonck et al. 2010), our wheelchair is designed to use a laser range finder as a main sensor for detecting obstacles because of its significant advantages such as providing accurate range data and high-resolution high speed streaming data, small size, and low power consumption (Hokuyo Automatic Co. 2005). Besides this laser, ultrasonic sensors and wheel encoders are also recruited in order to increase the reliability of the sensory information for the system. In terms of input methods, our system provides a number of different interfaces, from standard technology to hands-free technology, which match with the needs of a variety of users with different levels of injuries. For the elderly or people with low level injuries, joystick or iphone / ipad can be used to control the system. For those with severe impairments who have difficulty moving their upper limbs, then hands-

free technology such as head movement interface or brain computer interface would be the most appropriate interface for them to interact with the system. More importantly, regardless of the interface type, the user is always supported from the system to safely navigate in the unknown environment.

In terms of navigationally autonomous implementation, the learning machine paradigm is selected to avoid inherent shortcomings caused by route program paradigm as mentioned above. Our aim is to design an algorithm which is capable of effectively working in many unknown environment types and producing very reliable, smooth and optimal trajectories satisfying the requirements for such an assistive system. To make it clear, this research focuses on further improving an advanced learning technique first introduced in (Trieu, Nguyen & Willey 2007) that had demonstrated its ability to navigate throughout a dynamic environment in real time without collisions. Different to the previous work, we propose a more efficient method based on the Bayesian neural network in order to improve the performance of the autonomous algorithm, especially for difficult tasks like passing door or working in a narrow space.

In terms of shared control strategies, as exposed in this chapter, many different shared control mechanisms exist, depending on both the nature of user input signals and the designer's favourite style. Fully autonomous wheelchairs work in predetermined environments with very few user inputs. On the other hand, wheelchairs with full human control mode require a high degree of the user involvement, being continuous in case of manual control although they can operate in unknown environments. The last shared control strategy type, semi-autonomous strategy, acts like a combination of previous types in which both autonomous behaviours and user commands take part in the navigation process.

In this thesis, we will propose a new shared control principle which is able to adapt the degree of autonomy as the user wishes. The system does so by predicting the user's intent on the basis of the user's signal inputs and immediate environmental free space information at the decision time. However, unlike many other existing systems (LoPresti et al. 2011; Parikh et al. 2007) which are aimed towards users who are able to use the joystick

interface, or defined environmental types, our system is designed to work with people who have different levels of injuries as well as a variety of environments. Bayesian theory will again play a central role in combining the information from the environment and the user in order to make the most optimal shared control action for the system corresponding to each particular situation.

At the end of this project, we will arrange real-time validation experiments in which a number of participants are recruited to evaluate our designed system. In these experiments, both quantitative data and qualitative data are collected from the users and the systems. On the basis of the results, our work can be evaluated and compared to other systems. More importantly, the feedback from the human users and the system's real-time performance will be very important in order to adjust and upgrade to the better system.

Chapter 3

Real-time Free Space Determination Based on Bayesian Recursive Technique

3.1 Introduction

Chapter 2 provides a thoughtful literature review about a variety of intelligent wheelchair systems that have been developed around the world; advantages as well as disadvantages of those systems have also been exposed to form the gap for this research. Starting from this chapter, I will start to discuss in detail the methods we are developing for our system to overcome the drawbacks of other current systems.

The design in this study aims to develop an intelligent wheelchair that brings mobility back to elderly or disabled people in various unknown environments such as their home, hospitals, or rehabilitation centres. Therefore, the system will be built to operate in unknown environments relying solely on onboard sensory information. For that purpose, the data received from the sensors will be represented in a 2 – D map and be used to determine a number of available free spaces. Such information representation is necessary in order to find the direction of travel in combination with the user's commands, as well as to be employed as the inputs of the autonomous navigation algorithm in the next chapters.

For the system operating in dynamic environments, one of the most important issues associated with mapping is the uncertainty emanating from the contribution of many factors

such as unpredictable environments, the device limitations, etc. Usually, the uncertain factors cause considerable deterioration of the quality of measured map or free spaces. In this chapter, a new Bayesian probability free space construction method is proposed, which combines the information on the basis of uncertainty to make appropriate decisions for the wheelchair during navigation.

This chapter is presented as follows. Firstly, the advanced instrumentation that builds up our wheelchair system will be briefly introduced. Then, a method to integrate data from the sensors to effectively construct a 2-D real – time map is proposed. Finally, a Bayesian recursive technique will be applied to find the number of free space for the SAM wheelchair to travel.

3.2 Advanced Instrumentation

An intelligent wheelchair can be designed based on an actual mobile robot, or a commercially powered wheelchair. Considering advantages and disadvantages of each style, our SAM wheelchair has been designed from the commercially powered wheelchair. However, unlike other systems which heavily modify the underlying wheelchairs, the SAM wheelchair is built from add-on units that can be easily attached to and removed from the underlying wheelchair. The advantage of the add-on unit approach is that a customer is able to buy the system once and transfer it to multiple chairs over their lifetime. This approach also provides flexible configurations of sensors and the user interfaces depending upon each individual user's need.

The core in the SAM's design is a modification of the underlying power wheelchair that allows add-on units to be involved in the chair's control process. In particular, the SAM wheelchair is built around a commercial wheelchair – Invacare Roller M1 which has been adjusted to include the following items: a minicomputer, the user interfaces, and measurement sensors. Instead of using the joystick's signals to control wheel motors, control signals are generated from the onboard computer on the basis of an assistive navigation algorithm with inputs from the user interfaces and measurement sensors.

Although the SAM wheelchair with add-on unit approach is, in principle, capable of working with any type of user interfaces as well as sensors that can connect to the computer, this section only covers a connection of a limited number of user interfaces as well as sensors that are available at the Centre for Health Technologies, UTS. There are four user interfaces including joystick, head movement, brain computer interface, and iphone/ipad control user. The measurement sensors consist of a laser range finder sensor URG-04LX, two ultrasonic sensors, and two wheel encoders.

3.2.1 The Intelligent Wheelchair Design

3.2.1.1 Invacare Roller M1 with DX System

Before introducing our intelligent wheelchair's design, a brief of the underlying power wheelchair is needed to understand a later work. In this project, the power wheelchair Roller M1 is electrically controlled by DX system from Dynamic Controls, a global organization with corporate headquarters in New Zealand. Basically, the DX-system is a modular power chair control system. This modularity makes it possible to design a power chair that can meet the requirements of any user - requirements that can range from simple drive-only control to full environmental control. Just add additional modules when requirements grow.

The Roller M1 DX control system mainly consists of two parts as the figure 3.1: Joystick DX Master Remote (DX-RemG80A) and DX power module. The DX master remote is the brain of the DX system and is responsible for reading user commands, generating output commands, and sending these over the DX bus to the power module. The DX power module converts signals generated by the DX Remote into high current outputs provided from 24v battery. These outputs drive the wheel motors. See Appendix A for more details about power wheelchairs.

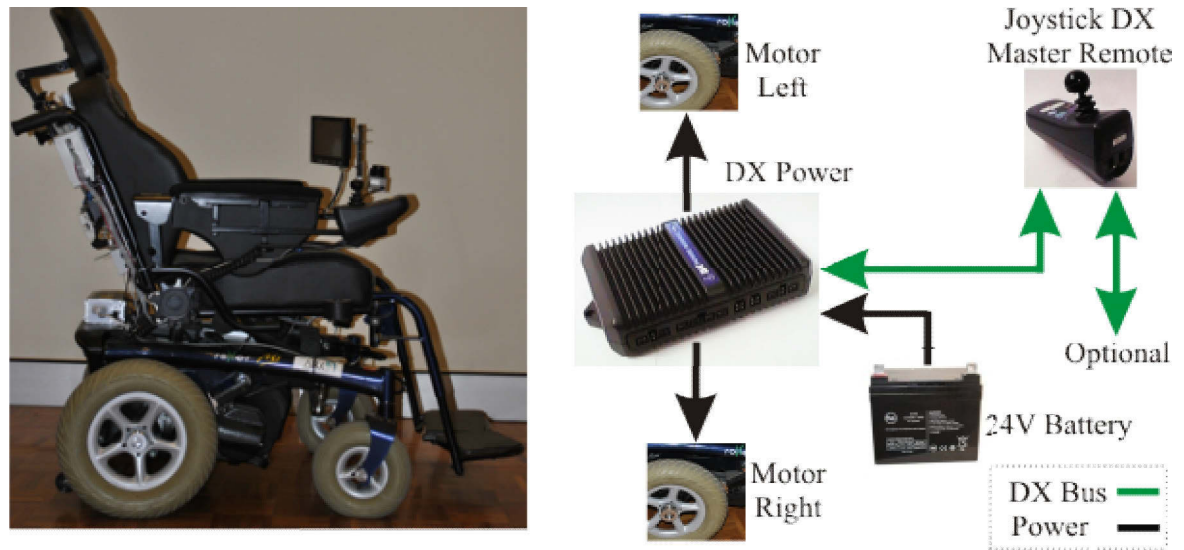


Figure 3.1: Invacare Roller M1 and electrical control diagram

All DX components are connected by DX bus based on CAN-Bus technology, and in the DX system, most devices have two DX bus connectors. The DX-RemG80A also has two DX bus connectors. One is used to communicate with the DX power module, and the other is optional and can be used to receive other input signals to control the system.

3.2.1.2 The Wheelchair Hardware System

Our intelligent wheelchair system is developed based on the DX system of the Invacare Roller M1. The power wheelchair has been modified to include the following items: a minicomputer, user interfaces, and measurement sensors. A hardware connection diagram of the intelligent wheelchair shown in the figure 3.2 consists of both add-on DX system components and add-on additional components designed at the Centre for Health Technologies, UTS.

The figure 3.3 illustrates the additional items that are attached on the front side as well as back side of the wheelchair Roller M1.

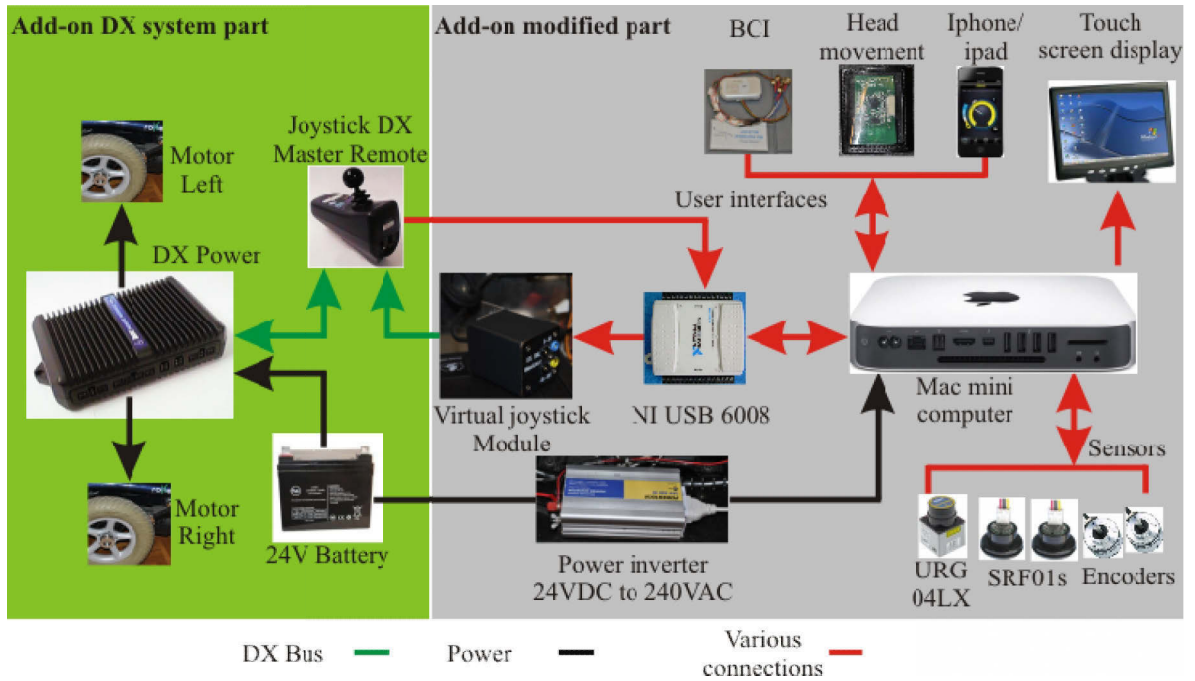


Figure 3.2: The intelligent wheelchair connection diagram

In the normal configuration, although the DX master remote has five modes that contain different parameters corresponding to different environments, the wheelchair is only controlled by the joystick of the DX master remote. In order to integrate the modified parts into the system as a whole, the DX master remote has been reprogrammed so that it can work with control signals received from both the joystick (as the normal wheelchair) and the external signals fed via an additional DX module called virtual joystick module. This modification accepts the user to operate the wheelchair as either a standard power wheelchair or the navigationally assistive wheelchair.

The driver simply uses the joystick to drive the vehicle as the normal operating mode is selected. On the contrary, in assistive mode operation, the on-board computer (Mac mini model A1347) powered from the wheelchair battery through the inverter 24VDC to 240VAC plays a central role in controlling the wheelchair. Instead of using the joystick, the on-board computer generates appropriate control signals following the calculations of assistive navigation software. These signals are transferred to the virtual joystick module via Ni USB 6008 (Instruments 2012) to control the wheelchair.

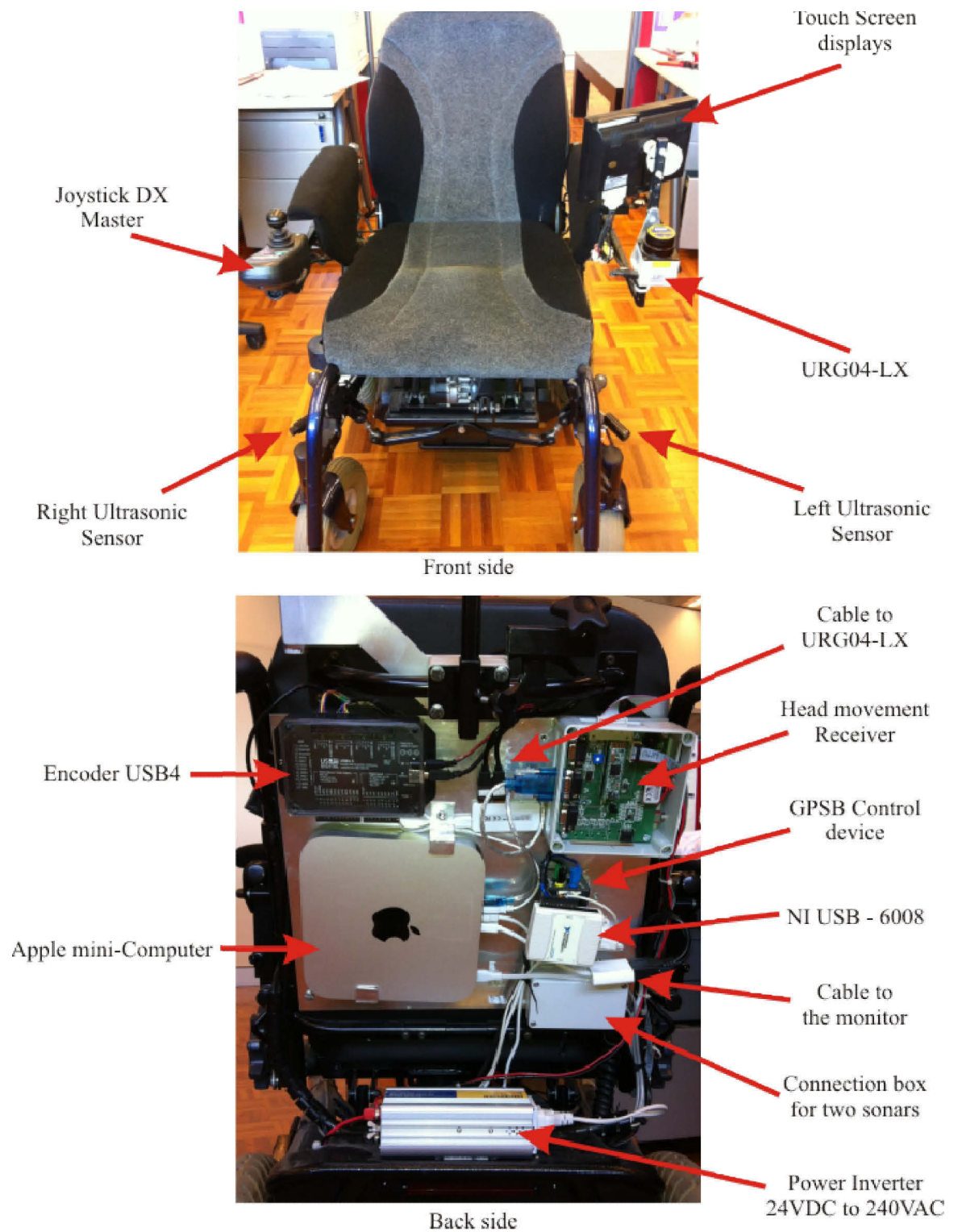


Figure 3.3: A prototype of the intelligent wheelchair

The navigation software is implemented to assist the user with lower and upper impairments with non-collision navigation. It does so by relying on both user commands and surrounding obstacle information. Currently, the user can employ one of four available interface types developed for the SAM wheelchair: joystick, head movement, brain computer interface (BCI), and iPhone4/iPad. The vehicle automatically detects and reads signals whenever these interfaces are connected to the system. Through these signals, the system is able to monitor what the user wants it to do.

Another important information source used to make navigation decisions comes from measurement sensors including a laser range finder URG-04LX, two ultrasonic sensors SRF01s, and two wheel encoders. The computer triggers the sensors to produce measurement information associated with the positions of obstacles surrounding the vehicle, as well as the translational and rotational speed of the vehicle.

3.2.1.3 The DX Master Remote Modification

The central part in the design of the SAM system is to modify in order to control the wheelchair by the onboard computer. The figure 3.4 shows how the computer is connected to the DX master remote. For the configuration, the DX master remote is programmed in order to receive either manual mode inputs from the joystick or the assistive mode inputs via the virtual joystick module. In terms of hardware, the virtual joystick module is simply a voltage receiver designed to accept the analog signals that are identical to those generated by a normal joystick, and transfer these signals to the DX master via a CAN bus compatible to the master module.

The USB-6008 is used to interface between the computer and the virtual joystick module. The digital signals from the computer will be converted into the analog forms and transferred to the virtual joystick via the pins of the USB – 6008. The device is also utilized to receive the joystick's signals via its analog input pins and send them to the on-board computer.

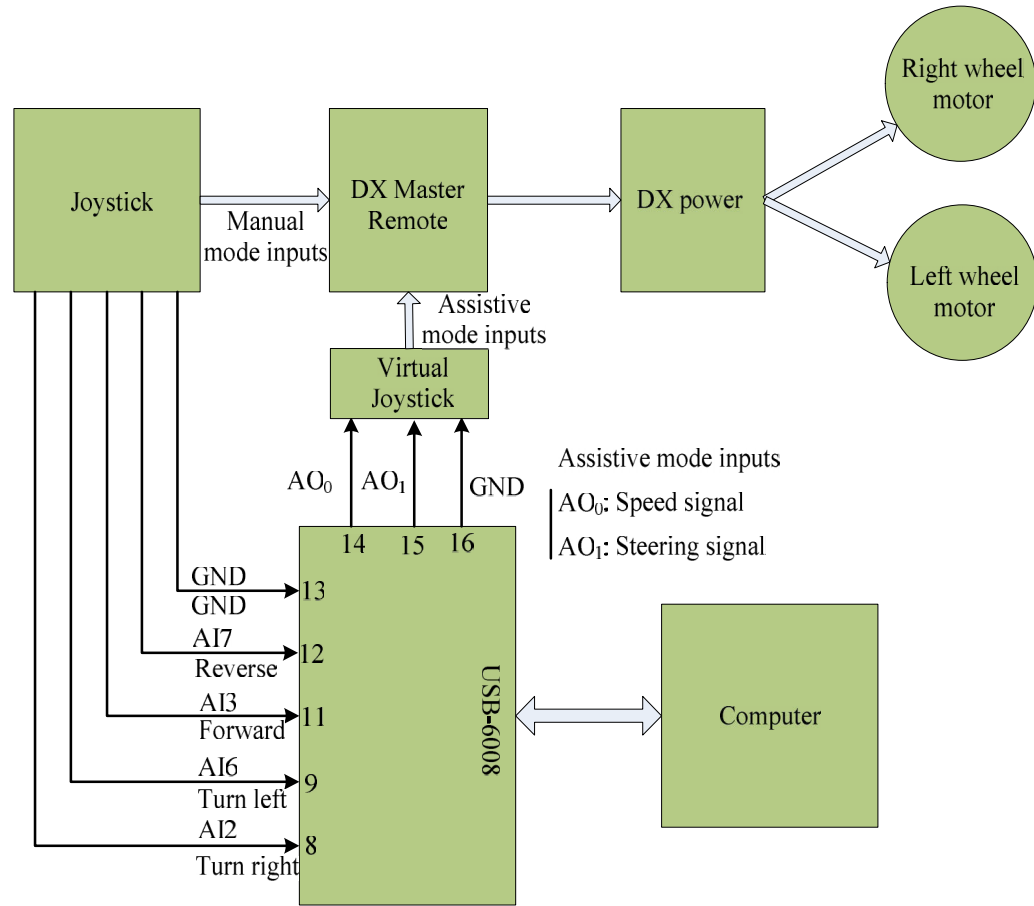


Figure 3.4: The Connection of the Onboard Computer to the DX Master Remote

3.2.1.4 Multi-Threading Based Software Design

The advent of new technology as well as competition between a number of operating systems such as Microsoft windows, GNU/Linux, or Mac OS have introduced new terms into the vocabulary of general-purpose PC users, like multithreading and multiprocessing.

Unlike multitasking which refers to the ability of operating systems that allow the execution of multiple tasks at one time, multithreading, however, extends the idea of multitasking to apply within applications, giving them the ability to separate their own tasks

into sub-tasks or individual threads. This technique brings many advantages enabling personal computers to implement high-performance real-time applications on multi-core systems.

In this research, the software for SAM is developed by using the multithreading approach in Labwindows/CVI 8.5, a product of National Instruments, based on Microsoft windows XP Service Pack 3 (SP3). By utilizing the approach, the overall wheelchair system will benefit from using CPU power more efficiently, improving the system reliability, and improving performance of the multiprocessor computer. The arguments for these advantages are discussed briefly as follows.

First of all, multithreading helps the system use CPU power more efficiently because it prevents synchronous acquisition calls from blocking other tasks. In the SAM wheelchair system, synchronous acquisition calls are used to communicate external devices, such as the laser URG-04LX, encoders, etc. In the single thread approach, these synchronous calls would prevent other tasks from executing until acquisitions were complete causing the processor to stand idle while the overall system is suspended. This block can be avoided by using the multithreading technique which assigns a synchronous call into one thread, thus, other parts of the program will still be executed rather than be stalled until the acquisition completes.

Secondly, by separating the program onto several threads, and also giving each thread an appropriate priority, we can prevent other operations in the program from adversely affecting important operations, thus increasing the reliability of the system. For example, updating data and providing safe trajectories for the wheelchair are much more important than displaying data on the screen, thus it would be more reasonable if the processor assigns a higher priority to updating data threads as well as calculating threads than that of displaying data thread.

Finally, Mac mini model A1347 with 2.66 GHz Core 2 Duo Processor is used as the central processor of our intelligent wheelchair. Using the multithreading technique makes several threads truly run simultaneously because each core can run a different thread at the same

time. Based on this, the system can obtain true parallel task execution which is impossible in the single-thread approach.

Using the multithreading approach in Labwindows/CVI 8.5, based on Microsoft windows XP Service Pack 3 (SP3), also supports symmetric multiprocessing (MSP) which allows the operating system to move threads between processors to balance the workload efficiently. The figure 3.5 presents SAM's software with five designed threads with automatic load balancing across multiple cores. Shared data between these threads is exchanged through a lock mechanism which prevents common data from accessing concurrently, including global variables, static local variables, and dynamically allocated memory that are accessible to all thread. Depending on the kinds of variables, one of following three-mechanisms would be used: thread locks, thread safe variables, and thread safe queues.

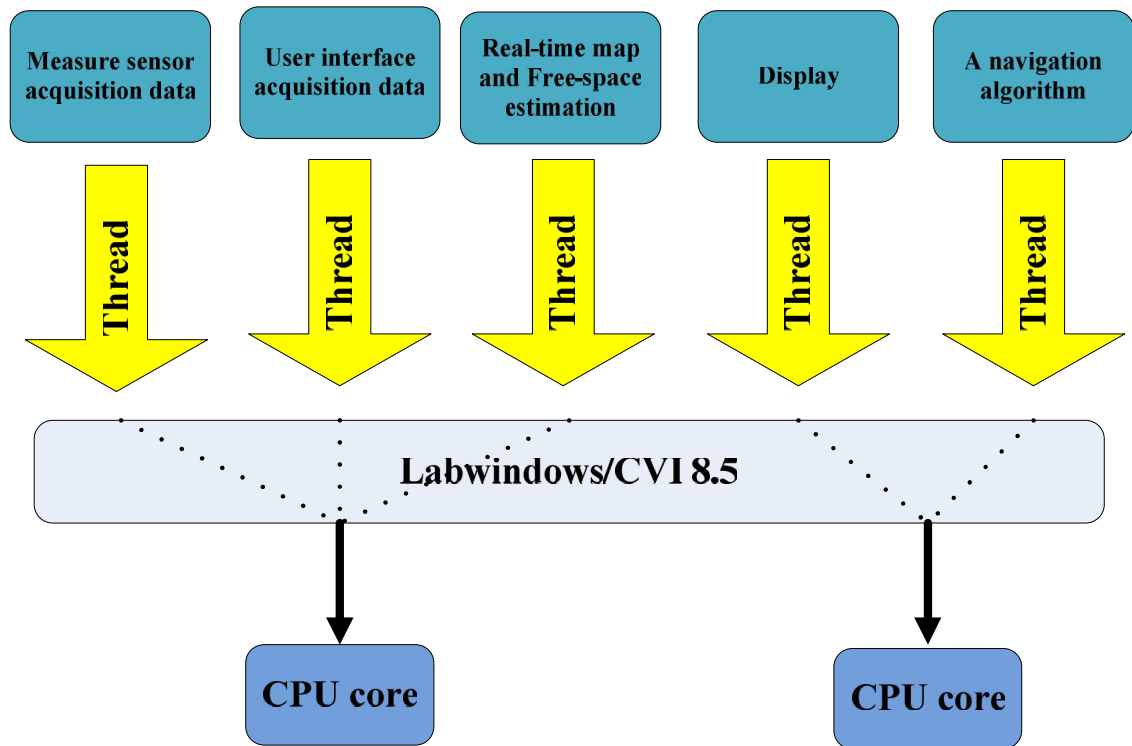


Figure 3.5: SAM's five threads with automatic load balancing across multiple cores

3.2.2 Sensors

The SAM wheelchair's measurement sensor system includes a laser range finder URG-04LX, two ultrasonic sensors SRF01s, and two wheel encoders. While the laser URG-04LX plays a central role in providing information about surrounding obstacles for non-collision navigation tasks, the two SRF01s attached on the right side and left side of the vehicle are used to detect transparent obstacles that are invisible to the laser URG – 04LX such as glass walls. The two wheel encoder sensors provide the system immediate speed as well as immediate heading direction of the vehicle.

3.2.2.1 Laser Range Finder URG - 04LX

Laser range finder URG-04LX is chosen as a main sensor for this system because it provides not only very accurate, but also high speed streaming data. Besides, the laser has significant advantages to a mobile vehicle project such as small dimensions, compact design and low power consumption.

The laser URG-04 is mounted at the left-front part of the vehicle, at a height of 0.75m. This sensor can operate up to 10Hz (scanning rate), with a 240^0 field – of – view, a 0.5^0 resolution (481 points), and maximum radius of 4m (Hokuyo Automatic Co. 2005). These specifications of the sensor are very appropriate for the proposed system as the sensor is able to give all information about the obstacles in front of the wheelchair. Based on the positions to the obstacles, the system can generate and execute a safe path (local path planning) while the user takes responsibility for global path planning.

In terms of communication to the on-board computer, the laser URG-04LX has a digital interface that supports both RS232C and USB 2.0. Although URG-04LX provides two types of physical connection to the host, both in fact work with a program written for RS232C, because USB has communication device class standard with its setting similar to RS232C. Based on its specifications, URG-04LX in the system is connected to the on-board computer via a USB cable. After installing the sensor's driver, the USB port would

appear as a virtual COM port. The protocol to communicate with the laser is identical to the industrial standard RS232.

The figure 3.6 shows steps that are programmed to work with URG-04LX, and the details of the programs can be seen in the appendix B.

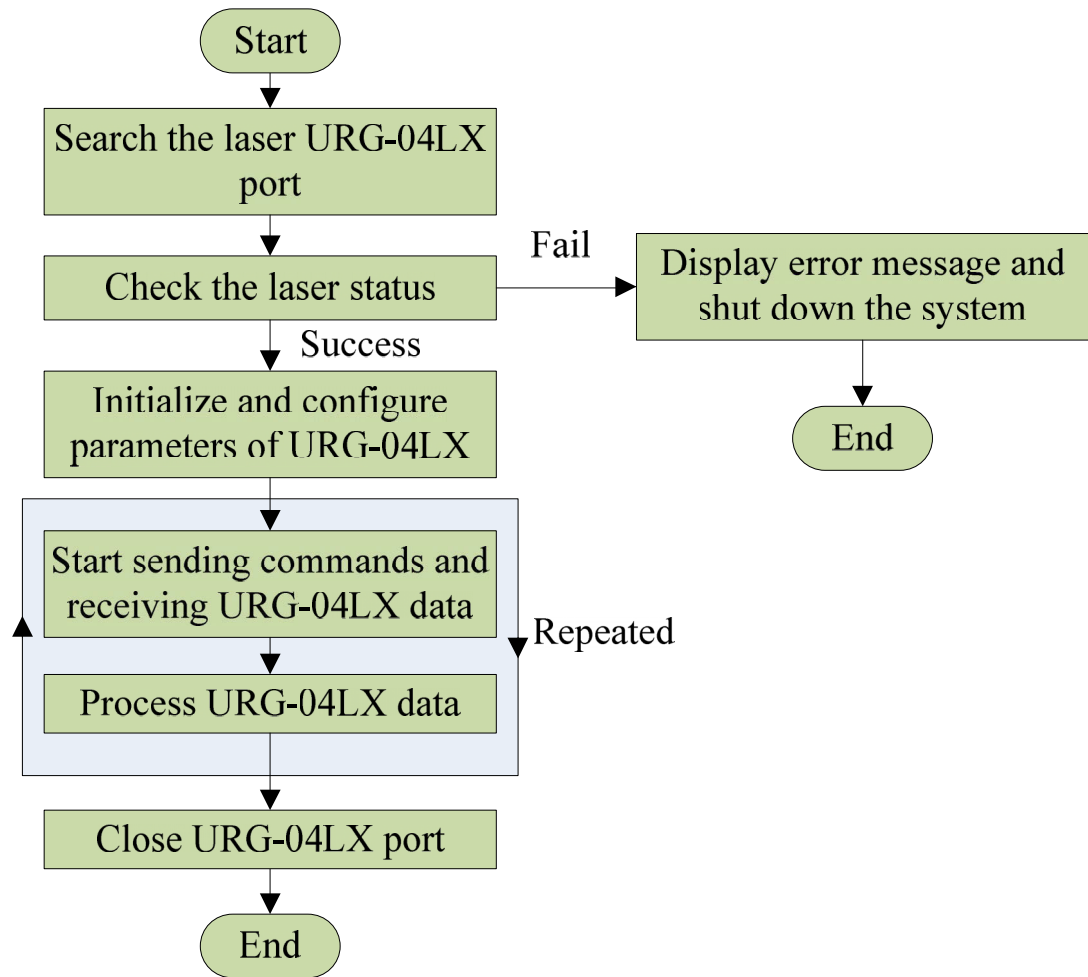


Figure 3.6: A flow chart of the program for the laser URG-04LX

3.2.2.2 Ultrasonic Sensor SRF01s

As URG-04 LX fails to detect transparent objects such as glass walls which are typically found in the home environment, it is necessary to equip SAM with another type of sensor that is able to supplement the URG-04LX's capabilities. Ultrasonic sensors are always the best solution for the situation because they are cheap and work very reliably with transparent obstacles.

Two SRF01s are used in the way that only provides supplementary information for the system. In other words, SRF01s measures obstacles on the right and left side of the vehicle. If the detected obstacles are in the range of URG-04LX, but are not in the obstacle data, then they would be added into the data to be involved in the computation process. The connection of both SRF01s to the system is identical and shown in the figure 3.7.

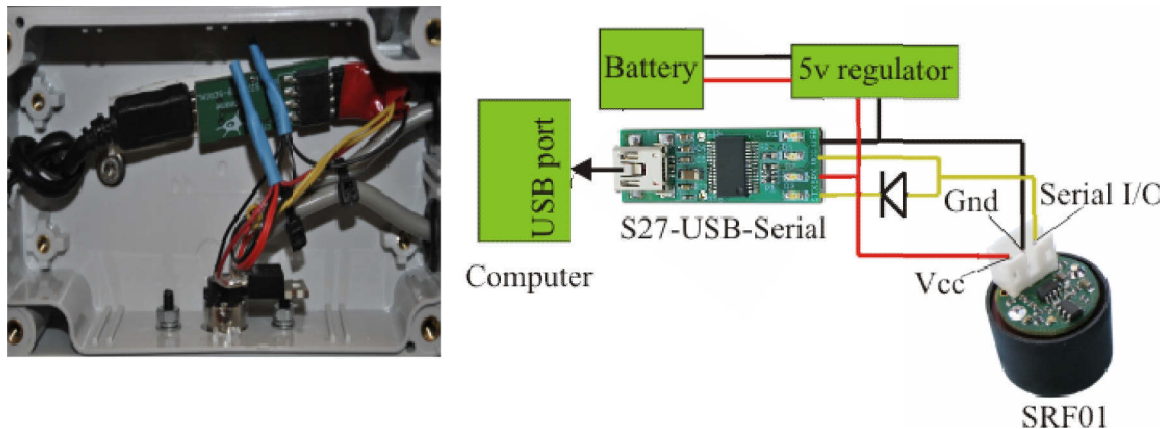


Figure 3.7: Connection of the Ultrasonic Sensor SRF01s to the System

In terms of the communication, the sensor's interface is a standard TTL level UART format, not RS 232(Robotics 2011). So, instead of plugging directly into the COM port which leads to damaging the sensors, the SRF01s are connected to the computer through S27-USB-Serial interface. After its driver is installed, this interface also appears as a virtual COM port and the way to communicate with the SRF01 is exactly like the way used to read information from a normal COM port.

The figure 3.8 illustrates a flow chart of the program written for the SRF01s, and the details about this program can be found in the appendix B.

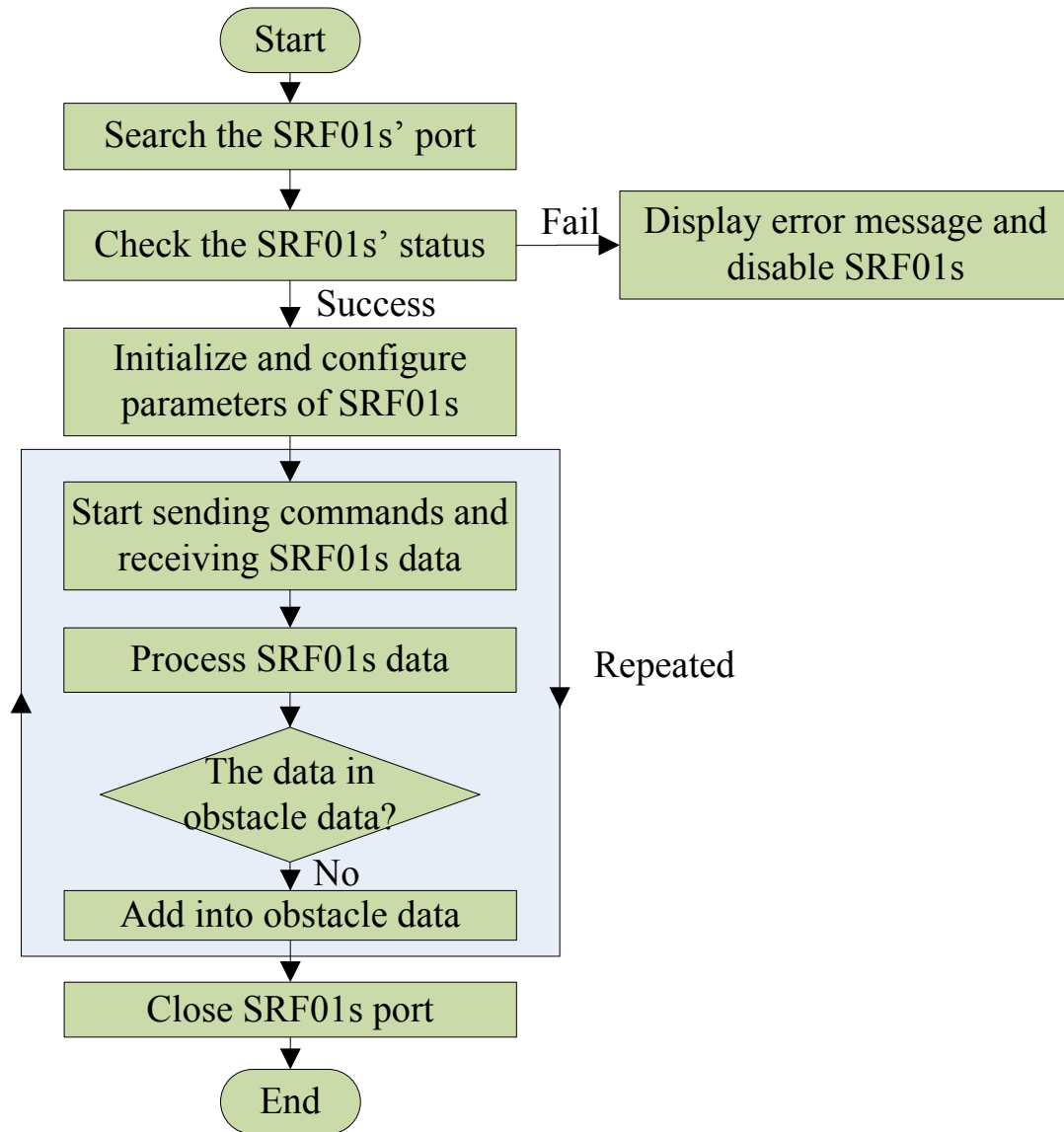


Figure 3.8: A flow chart of the program for SRF01s

3.2.2.3 Encoders

The Invacare Roller M1 has a “differential drive” design that uses two independently powered wheels to move and steer. Thus, in order to measure speed and steering angle of the vehicle, two shaft encoders have been attached into two powered wheels. An Encoder data acquisition USB device, USB4-S, a product of US digital, is used to manage these

encoders' data. It commands encoders to collect data, processes the data, and then transfers the data to the on-board computer through a USB port. The figure 3.9 provides a

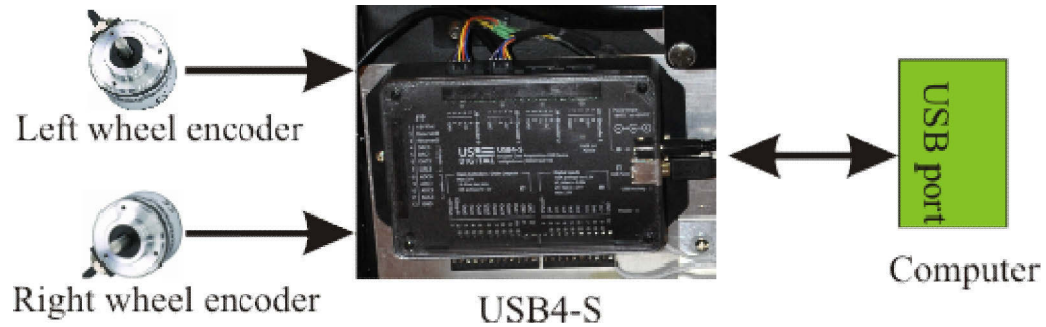


Figure 3.9: Connection of wheel encoders to the system

connection of encoders, USB4-S, and the computer.

Because the data from USB4-S is only a counter (encoder ticks) indicating the angular position of the drive wheel, dead reckoning technique is recruited to determine the straight speed and the steering angle of the wheelchair. Relative positioning based on the dead reckoning has been widely employed in most mobile robot applications for some time now because it is simple, inexpensive, and easy to accomplish in real-time. Basically, this method simply monitors the wheel revolutions to compute the offset from a known starting point. Despite advantages of dead reckoning, this approach is severely impaired from its unbounded accumulation of errors if the travel position estimate is as little as 10m.

Although typical dead reckoning errors become unacceptably considerable with the travel position estimation, they are certainly ignored for instant measures such as the current speed or the current heading direction of the vehicle. These instant values are understood in the way that the change in measure values between two consecutive reading data intervals is recorded. In the SAM system, dead reckoning is used to estimate both instant straight movement and instant heading of direction that partly provide a description of the dynamics of the vehicle for calculations later.

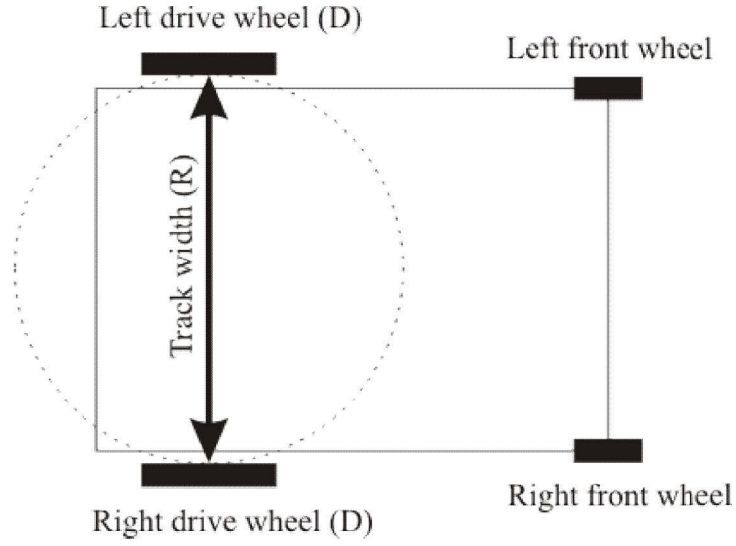


Figure 3.10: The Geometry of the SAM

Based on the geometry of the SAM wheelchair in figure 3.10, the instant movement (ΔS) and steering angle ($\Delta\alpha$) at i^{th} computational cycle can be respectively estimated by:

$$\Delta S = \left[\frac{\Delta N_{\text{left}} + \Delta N_{\text{right}}}{2} \right] \times \frac{\pi D}{N} \quad (3.1)$$

$$\Delta\alpha = \left[\Delta N_{\text{left}} - \Delta N_{\text{right}} \right] \times \frac{\pi D}{RN} \quad (3.2)$$

where: D: The wheelchair diameter

R: track width.

ΔN_{left} , ΔN_{right} are the incremental count of the left and the right wheelchair encoder between two consecutive cycles.

N: is the resolution of the encoders.

The figure 3.11 illustrates a flow chart of the program written for dead reckoning, and the details about this program can be found in appendix B.

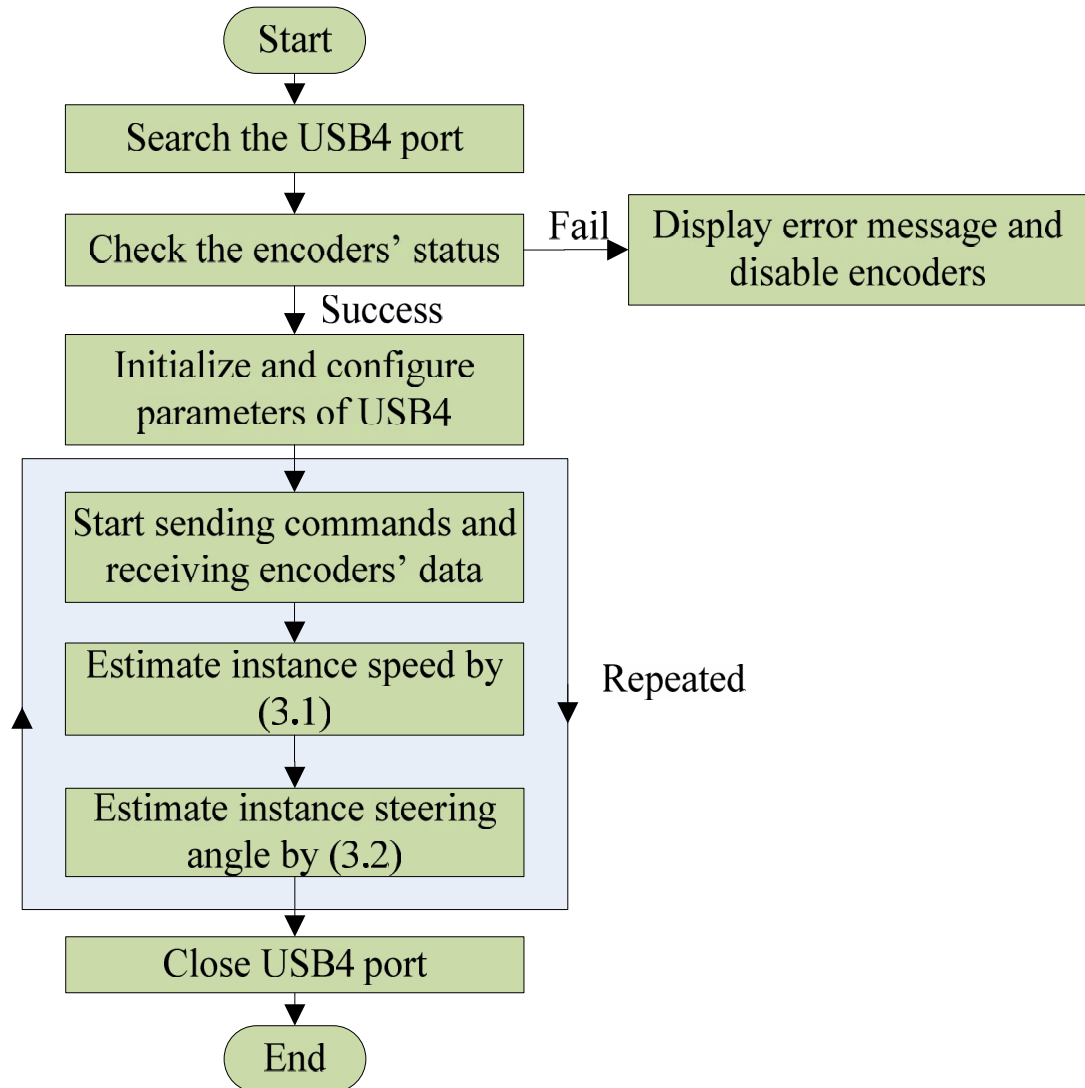


Figure 3.11: A flow chart of the program for the encoders

3.2.3 The Human – Machine Interfaces

The SAM wheelchair system has been used to explore a variety of alternatives to the more traditional user interfaces associated with power wheelchairs. Unlike standard wheelchairs, the SAM wheelchair has an on-board computer with which many input sensors can interface. More importantly, assistive navigation software provided by the system provides a safety net for user interfaces that are inaccurate or have limited bandwidth. Besides a standard joystick, other exotic user interfaces have been implemented for the vehicle.

3.2.3.1 Joystick

The joystick is a hands controlled sensor that produces proportional signals referring to the direction and turning angle of the wheelchair. In the SAM design, the joystick was connected to the computer via the NI USB 6008. Two signal channels X and Y from the joystick pins were recorded and displayed as the figure 3.12. As depicted, the electrical values from the channels were changed when the joystick was moved into different positions. The Y signal usually changes significantly when the joystick was moved forward or backward ranging from $\approx 3.8\text{V}$ when the joystick completely pointed to forward and $\approx 1.2\text{V}$ when joystick pointed completely to reverse. Similarly, the X signal ranges from $\approx 3.8\text{V}$ as the joystick completely points to right-side to $\approx 1.2\text{V}$ as the joystick points completely to left-side.

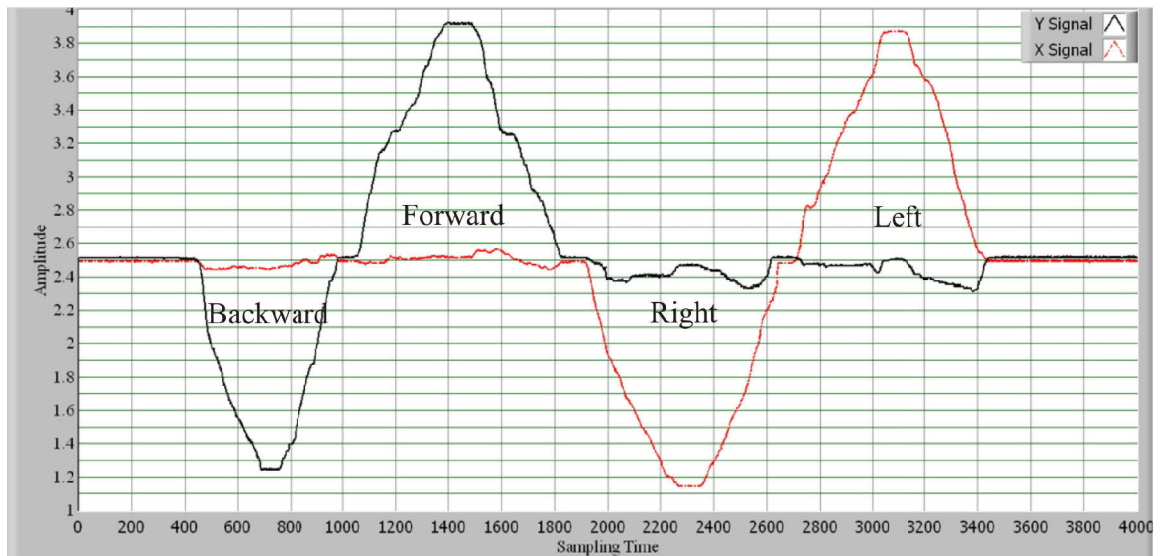


Figure 3.12: The voltage generated by the joystick sensor with different positions

Obviously, these signals can be directly used to control the wheelchair manually. However, for the users with disability, in many cases, they are unable to control the joystick properly or they are too tired to involve the navigation process, and therefore, the signals produced by the users are often inconsistent, and inappropriate to control the wheel motors. For this reason, the system should real-time monitor and process the joystick's signals to determine the control command that describes a general direction of travel which the user would like to go through. The control command might be one of following commands: go forward,

turn left, turn right, and stop. It should be noted that the “stop” control command is defined in case the joystick is moved backward.

3.2.3.2 Iphone/ipad Interface

An idea to integrate iphone or ipad with wheelchairs is fantastic in terms of enabling a user to enjoy all benefits of a device which has many features such as writing emails, surfing the web, video chatting with family and friends, etc. At the Centre for Health Technologies - UTS, an iPhone / iPad interface has been developed for users to display information and control the intelligent wheelchair as shown in the figure 3.13.

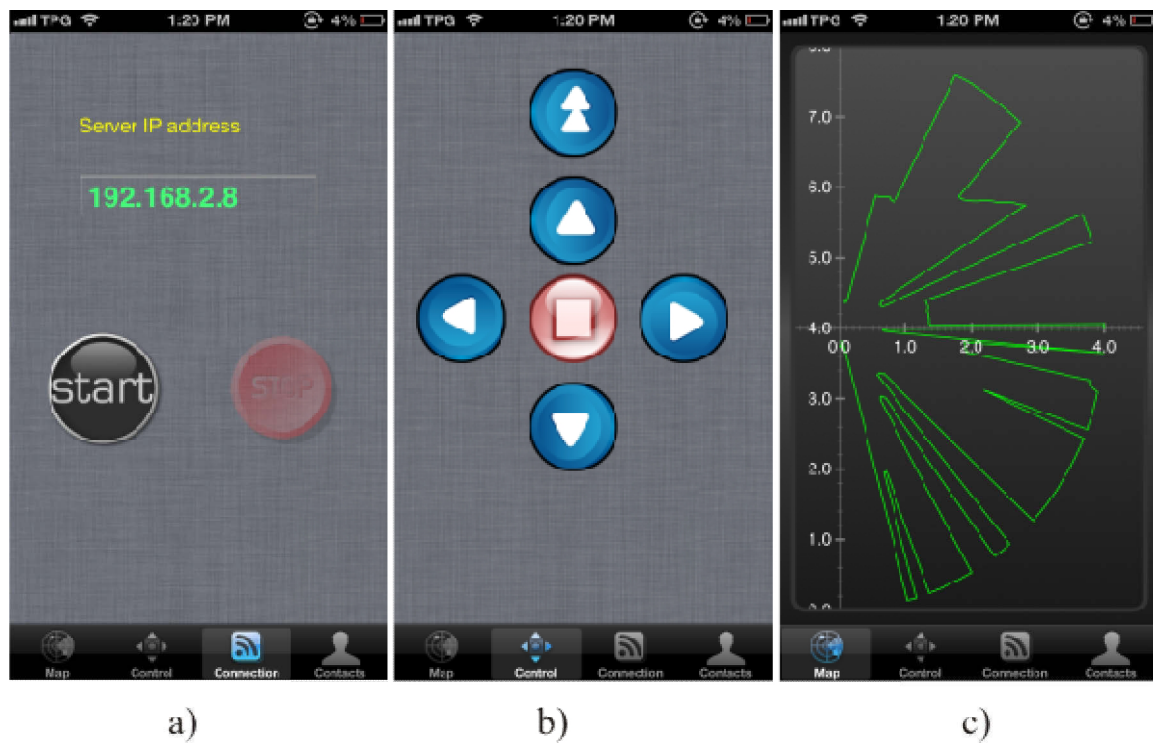


Figure 3.13: The iPhone/iPad interface for SAM

The operating principle is that the iPhone / iPad and the wheelchair onboard computer are wirelessly connected together via a building internet router. The wheelchair system identifies the iPhone/iPad by its IP address as the figure 3.13 a). This means the interface would be able to control the wheelchair from anywhere as long as it is connected to the

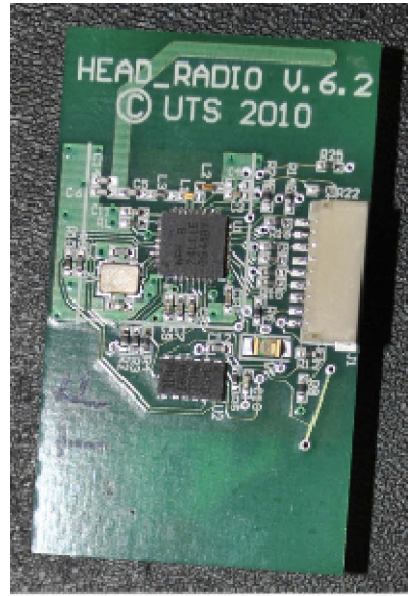
internet. This is very helpful because the user's relative or caregiver can help to drive the system out of some difficult situations remotely.

The user can control the system as well as monitor the real-time map display via the functionality windows of the developed app as depicted in the figure 3.13 b) and c). With the control window, there are six actions which the user can take to communicate with the wheelchair. Along with four commands corresponding to 4 directions (forward, right, left, backward), an increase speed button is provided to increase the movement of the wheelchair. The backward and increase speed button is disabled when the wheelchair is operating at the assistive mode. The display window is to represent the 2-D real-time map and the computing direction of travel uploaded from the onboard computer in real time. This function assists the user to supervise and realize the perception of the wheelchair, this is especially important as the user controls the SAM wheelchair remotely.

3.2.3.3 Head-Movement Interface

The head movement interface consists of two separated parts: two accelerometers and a receiver as shown in the figure 3.14. One accelerometer – the reference sensor - attached behind the chair plays a role as the reference, and another accelerometer - the head sensor - is attached on the user's head to control the wheelchair and is connected to the receiver board via the radio signal. The receiver is attached at the back of the wheelchair to process data received from the two accelerometers, and then it transfers the data to the onboard computer through a Prolific USB-to-Serial/UART Port.

Data received from the head movement include the X and Y tilt angle differences between the wheelchair detected by the reference accelerometer behind the chair and the head accelerometer. The X tilt controls the forward and backward speed and the Y tilt control left and right turning. A tilt of about 10 to 40 degree produces a speed range of zero and to maximum. This is a direct measurement of the degree of tilt of the head.



A head sensor circuit



A head sensor receiver

Figure 3.14: The head movement interface parts

In chapter 5, the details about processing the head movement data will be presented, in which the movement of the user's head will be converted into analog signals. Similar to the joystick, the voltage values of the head-movement interface after rescaling can be directly used to control the wheelchair manually by the able users. However, for impaired users, it is impossible to manually control the wheelchair because of inconsistent and noisy signals produced by the target users. A Bayesian neural network will be proposed in order to find the user control command for the head movement interface.

3.2.3.4 Brain – Computer Interface

People with severe motor disabilities, such as very high level spinal cord injury, amyotrophic lateral sclerosis, and brainstem stroke, have few means to communicate with the outside world. Brain computer interface (BCI) technology is intended to establish direct communication between the brain and external devices to provide effective communication and to eventually restore volitional movement.

The current approach is to extract control signals from recording neural activity by inserting microelectrodes into the cortex. This approach is invasive, and there are barriers to the long-term stability of the recorded signals. Electroencephalography (EEG) is non-invasive, but it has very limited information content due to the low-pass filter properties of the skull. A high-performance BCI system could bring substantial clinical benefit for patients with motor disabilities without significant additional risk.

At the Centre for Health Technologies, a wireless system to acquire wireless EEG signals in real-time has been developed as shown in the figure 3.15.



Figure 3.15: The wireless EEG hardware interface

It consists of a Tx transmitter headset module and an Rx transceiver module. In the transmitter module, there are two differential EEG channels, a PIC micro-controller and a 2.4 GHz transceiver. The Rx transceiver module consists of a 2.4 GHz transceiver for two channels, a PIC micro-controller and a USB communication to a PC or microcontroller. The USB receiver uses a virtual COM port for transfer to the host with the baud rate fixed

at 115,200 bps. The bandwidth of the EEG system is from 0.1 Hz to 150 Hz and the positions of EEG sensors are placed at O1, O2, P3, and P4. The sample rate is 256 Hz and 2-second epochs are spectrally analysed into four frequency bands for analysis, delta (0.5-3.0 Hz), theta (3.5-7.5 Hz), alpha (8-13 Hz) and beta (13.5-45 Hz). An automatic real-time process is used to eliminate EEG epochs which are contaminated by artefacts.

The EEG signals, after being received and processed, give the system one of five commands: Forward, turn left, turn right, and backward. The figure 3.16 shows the interface of the EEF processing program.

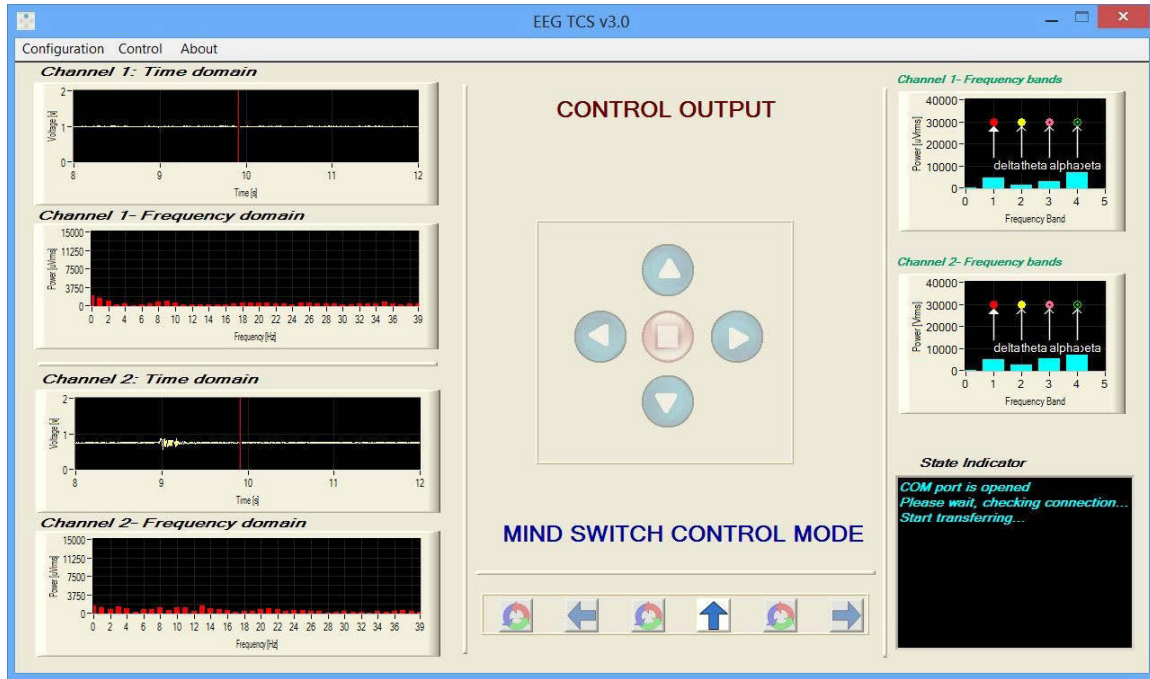


Figure 3.16: The brain computer interface for SAM

3.3 Free Space Determination Using Bayesian Recursive Technique

The main goal in designing our wheelchair is to assist a disabled user to navigate in the unknown and dense environment. Because the user with impairments is generally unable to

continuously navigate the wheelchair, this vehicle, therefore, must be capable of planning and executing motion tasks autonomously to support the user. In terms of navigation, to reach a reasonable degree of autonomy, two basic requirements are needed: sensing and reasoning. The first one is provided by an on-board sensory system that gathers local information about the wheelchair itself and the surrounding scene. The second one is accomplished by devising algorithms that exploit this information in order to generate appropriate commands for the wheelchair. To make the use of the wheelchair appealing in real-time applications, it is necessary to call for efficient algorithms that can guarantee real-time performance in the presence of insufficient or conflicting data.

This section includes two sub-sections: Real-time map and free space determination. The next section presents the method of building a 2D local map by combining information sources from available sensors. The free space determination will introduce a simple but effective technique to find accessible spaces for the navigation.

3.3.1 Real-time Map

In this section, a 2-D real-time map will be built on the basis of combining information sources obtained from two types of sensors in the SAM's design: A laser range finder URG-04LX and two wheel encoders. These sensors are requested to repeatedly update new data with the rate of data acquisition at 10Hz. This updating speed is fast enough to provide real-time data for the SAM system to navigate safely in dynamic environments.

The detected obstacles in this map are represented by their positions in relation to the origin or a reference point which is defined as the middle of the driving wheel shaft. As discussed in the section 3.2.2.1, the laser URG-04LX is mounted on the left arm rest, so it is worth noticing that due to the difference of the position of the laser and the reference point in the system design, it is necessary to move the laser's data into the reference point before any calculation has taken place by the following formula.

$$\begin{cases} X_{new} = X + X_{Laser} \\ Y_{new} = Y + Y_{Laser} \end{cases} \quad (3.3)$$

where (X,Y) are coordinates of an obstacle with the origin attached on the laser, and (X_{new},Y_{new}) are the new coordinates of the obstacle moved into the reference point. (X_{laser}, Y_{new}) are the coordinates of the URG-04LX to the reference point.

Currently, the SAM is designed to move upwards, and thus, the real-time map should provide enough information about the obstacles related to the navigation process. The sensor URG-04LX is iterated to scan obstacles with the begin angle 0^0 to the end angle 180^0 , the visible field of the laser UGR-04LX only covers a green-blue semi-circle area as shown in the figure 3.17 a). However, due to the location of this sensor in front of the wheelchair, there is an area near the right and left side of the wheelchair that is not detected by this sensor, and that might cause an unexpected collision. In other words, the visible area provided by the URG-04LX is not enough to guarantee a safe navigation.

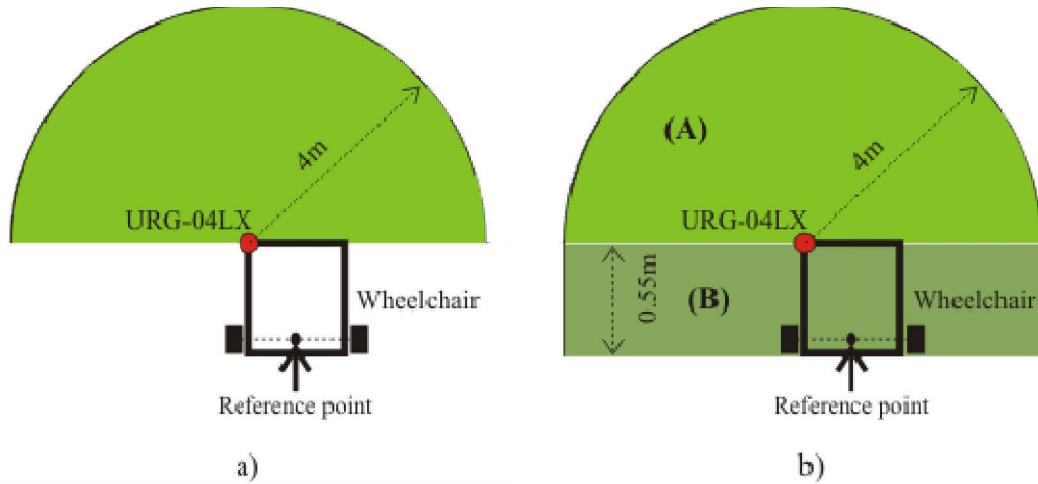


Figure 3.17: The visible field of the wheelchair (not in scale)
a) The visible field of URG-04LX b) Visible field of the Real-time map

In order to obtain a fully visible area in the navigational direction, we propose that the SAM utilizes both data sources from the laser URG-04LX and the wheel encoder system. In this proposal, the obstacles in the undetected part as just mentioned can be estimated by combining their data based on a technique called a relative positioning method. In particular, the SAM's real-time map is comprised of two main parts as depicted in the figure 3.17b). The first part (A) is directly provided by the laser range finder URG-04LX,

and the second one (B) is indirectly determined by a relative positioning method integrating the data from the laser in the previous cycles and the encoders' data.

The principle to construct area (B) is demonstrated in the figure 3.18. Consider two consecutive computational cycles (n-1) and n. In the cycle (n-1), the laser sensor detected an obstacle (a) in front of the wheelchair; coordinates of the obstacle (X_{an-1} , Y_{an-1}) were recorded in the system. Next cycle n, the obstacle (a) is no longer observed by the sensor as the wheelchair moved a distance determined by a triplet (ΔX_n , ΔY_n , $\Delta \theta_n$), and the obstacle (a) already lies beyond the scope of the sensor URG-04LX. The coordinate of the obstacle (a) can be estimated in the new position of the wheelchair by the following equation:

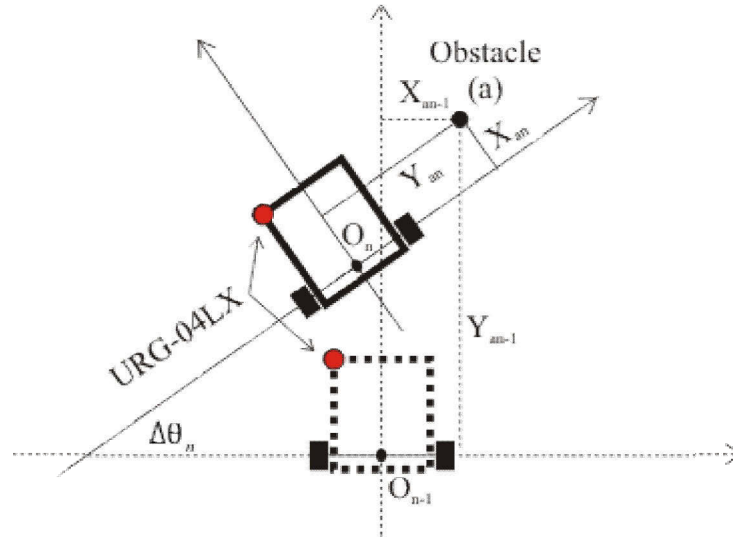


Figure 3.18: The visible field of the wheelchair map

$$\begin{cases} X_{an} = (X_{an-1} - \Delta X_n) \cos(\Delta \theta_n) + (Y_{an-1} - \Delta Y_n) \sin(\Delta \theta_n) \\ Y_{an} = -(X_{an-1} - \Delta X_n) \sin(\Delta \theta_n) + (Y_{an-1} - \Delta Y_n) \cos(\Delta \theta_n) \end{cases} \quad (3.4)$$

where

X_{an} , Y_{an} : The position of the obstacle a in the new coordinates $O_n X_n Y_n$

X_{an-1} , Y_{an-1} : The position of the obstacle a in the previous coordinates $O_{n-1} X_{n-1} Y_{n-1}$

$(\Delta X_n, \Delta Y_n, \Delta \theta_n)$: The triplet that determines the new position of the wheelchair with respect to the previous location.

As discussed in the section 3.2.2.3, the movement of the wheelchair is detected by the encoder system in terms of distance travelled, ΔS , and the angle change, $\Delta\theta$. From these two parameters, a location estimation method (Wang 1988) has been proposed to find the triplet of the new position of the wheelchair.

$$\begin{cases} X_n = X_{n-1} + \Delta X_n \approx X_{n-1} + \Delta S \cos(\theta_{n-1} + \Delta\theta_n / 2) \sin(\Delta\theta_n / 2) / (\Delta\theta_n / 2) \\ Y_n = Y_{n-1} + \Delta Y_n \approx Y_{n-1} + \Delta S \sin(\theta_{n-1} + \Delta\theta_n / 2) \sin(\Delta\theta_n / 2) / (\Delta\theta_n / 2) \\ \theta_n = \theta_{n-1} + \Delta\theta_n \end{cases} \quad (3.5)$$

Because the new position is calculated with respect to the previous location, so let $X_{n-1} = 0$, $Y_{n-1} = 0$, and $\theta_{n-1} = \pi/2$, we can simplify the equation (3.5) by

$$\begin{cases} \Delta X_n = -\Delta S \sin(\Delta\theta_n / 2) \sin(\Delta\theta_n / 2) / (\Delta\theta_n / 2) \\ \Delta Y_n = \Delta S \cos(\Delta\theta_n / 2) \sin(\Delta\theta_n / 2) / (\Delta\theta_n / 2) \end{cases} \quad (3.6)$$

Combine the equations (3.4), (3.5), and (3.6), we can find the position of the obstacles that lie in the region B.

Until now, we have mostly built a fully visible area map covering around the wheelchair. Nevertheless, from a practical point of view, the map data would be better rearranged into a standard form. For instance, a number of the map data points need to be fixed, and this number should be chosen so that it not only saves the memory (very important for global variables in the multi-threading software), but still contains sufficient information. In practice, we can transform the map into 180 data points, then save them into an array. By doing so, we only use one array to represent all obstacle information around the wheelchair in a polar form because an array element value indicates the obstacle distance to the wheelchair, while its index shows the obstacle angle.

Finally, the mapping algorithm is summarized as follows:

1. Received data from URG-04LX and the wheel encoders.
2. Initialize numerical arrays containing the mapping data.
3. Move the map data in the previous iteration by the equation (3.6).

4. Move the URG-04 data into the preference point using the equation (3.3).
5. Replace a part of map data that belongs to the region (A) by the moved URG-04LX data ($Y_{\text{new}} > Y_{\text{laser}}$)
6. Remove a part of the map data with $Y_{\text{new}} < 0$
7. Rearrange the map data into a 180 data point array

The details about this program can be seen in appendix B

3.3.2 Free Space Determination Using Bayesian Recursive Technique

In addition to the real-time map, it is necessary to determine free spaces for the wheelchair to navigate without collisions. There are many methods that have been reported to find free spaces or potential paths for circumnavigation in the literature. Despite various techniques, most of them rely on an assumption that the vehicle acts as a particle or a point which was originally introduced by (Lozano-Perez 1983). This classic method assumes that the vehicle can be represented as a point and is capable of omnidirectional motion. In the reports (Trieu, Nguyen & Willey 2007, 2008a), this principle was applied for our SAM system. To treat it as a point, the wheelchair is shrunk as a single point by enlarging obstacles according to the wheelchair's dimension, a safe distance and its orientation. The advantage of this method is easy to implement.

However, extending an obstacle makes narrow paths such as doorways become a challenge. As illustrated in figure 3.19.a), a doorway that is actually traversable by the wheelchair is blocked by extended obstacles. Moreover, the pragmatic method is unable to deal with errors caused by the measurement noise as well as taking into account the capability of the wheelchair passing the door. As a result, the system even fails to pass through paths which have been considered as free spaces.

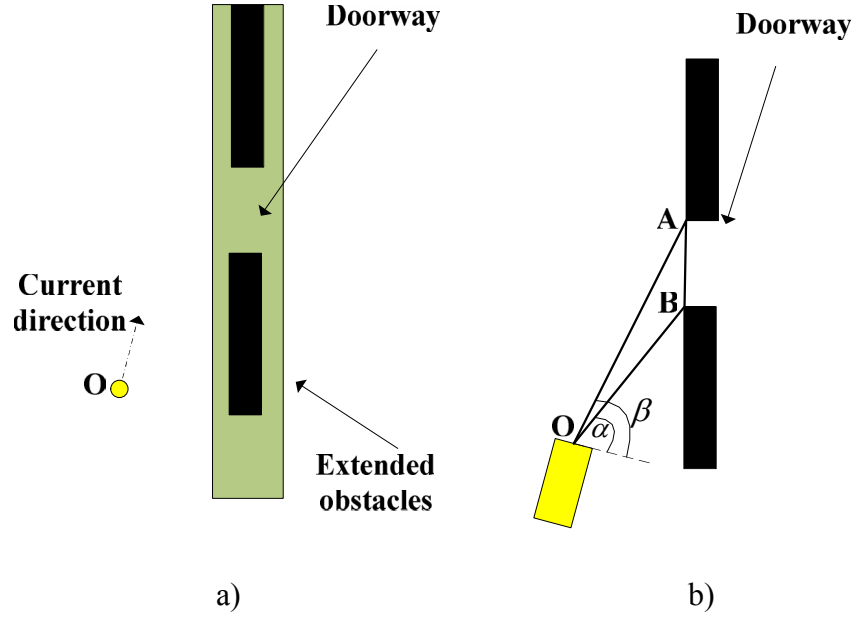


Figure 3.19: The wheelchair with a narrow opening
a) The wheelchair at traversable doorway blocked by extended obstacles
b) Determining a potential path based on the path's size

The fact is that a door can be considered as a free space if the wheelchair is capable of passing through it. Otherwise, it should be considered as an obstacle for the system. Generally, passing through a door, especially in narrow one, is usually quite uncertain for an autonomous system. The uncertainty rises from the limitation of its hardware as well as the autonomous navigation algorithm used by the system. Because of those reasons, the appropriate method should provide free space determination based on the uncertainty facts, and then the system makes its optimal decisions from the probability of the free space calculation.

This chapter will propose an advanced technique based on Bayesian recursive to determine free spaces for the SAM wheelchair. First, a limited number of opening spaces are considered and compared with the wheelchair's dimensions to see whether they are suitable for potential travel candidates or not. As shown in the figure 3.19.b, the wheelchair measures the size of the doorway opening by the following equation.

$$d = AB = \sqrt{OA^2 + OB^2 - 2OAOB \cos(\beta - \alpha)} \quad (3.7)$$

If the size is greater than the wheelchair's width size plus a safe distance, then it is considered as one of the potential free spaces of travel, if not, it might be an obstacle.

Next, the initial numerical judgment of the path is modelled into the probabilistic information on the views of the sensor's measurement errors and the capability of passing the door of the system. Finally, the probabilistic information sources are combined using the Bayesian theory to find the probability of the considered free space. These steps of this algorithm are recursively applied to provide the free space determination in the real-time for the system.

Let's assume that the states of a door up to the time of measuring t are presented as

$$W_{1:t} = \{w_1, w_2, \dots, w_t\} \quad (3.8)$$

In which, the state w_t can be one of the two following states

$$w_t = \begin{cases} FS & \text{a free - space for SAM} \\ OB & \text{an obstacle for SAM} \end{cases} \quad (3.9)$$

For generalization, let's further assume that based on the state w_t , the SAM wheelchair would make its own control decision u_t which might be one of two following actions

$$u_t = \begin{cases} move & \text{for SAM to move the free - space} \\ stop & \text{for SAM to stop} \end{cases} \quad (3.10)$$

The state w_t of the door at time t can be determined from the sensor's noisy measurement $Z_t = \{z_1, z_2, \dots, z_t\}$, and a set of the command variables $U_{1:t} = \{u_1, u_2, \dots, u_t\}$ via the Bayesian rule as follows

$$P(w_t | Z_{1:t}, U_{1:t}) = \frac{P(z_t | w_t, Z_{1:t-1}, U_{1:t}) P(w_t | Z_{1:t-1}, U_{1:t})}{P(z_t | Z_{1:t-1}, U_{1:t})} \quad (3.11)$$

As the probability $P(z_t | Z_{1:t-1}, U_{1:t})$ is independent of the state w_t of the door, then it can be set as a normalized constant η , thus, the equation (3.11) can be rewritten as

$$P(w_t | Z_{1:t}, U_{1:t}) = \eta P(z_t | w_t, Z_{1:t-1}, U_{1:t}) P(w_t | Z_{1:t-1}, U_{1:t}) \quad (3.12)$$

From the equation (3.12), the probability $P(w_t | Z_{1:t}, U_{1:t})$ of the state w_t of the considering door at time t would be estimated by the probability $P(w_t | Z_{1:t-1}, U_{1:t})$ and the probability of $P(z_t | w_t, Z_{1:t-1}, U_{1:t})$. The method to update the probability of the state w_t , therefore, involves two essential steps which are called the prediction and the measurement update as follows.

In the prediction step, the SAM wheelchair predicts the probability of the state w_t if it makes the action u_t . In other words, the new command u_t is processed to find the prediction probability $P(w_t | Z_{1:t-1}, U_{1:t})$ from the prior probability $P(w_{t-1} | Z_{1:t-1}, U_{1:t-1})$ of the previous state.

$$\begin{aligned} P_c(w_t) &= P(w_t | Z_{1:t-1}, U_{1:t}) \\ &= \int P(w_t | w_{t-1}, Z_{1:t-1}, U_{1:t}) P(w_{t-1} | Z_{1:t-1}, U_{1:t-1}) dw_{t-1} \end{aligned} \quad (3.13)$$

Apply the Markov assumption that the state w_t is to be complete, the state w_{t-1} , past measurements contain no information regarding to the state w_t , and the state w_{t-1} does not depend on the future command u_t . The equation (3.13) can be safely rewritten as

$$P_c(w_t) = \int P(w_t | w_{t-1}, u_t) P(w_{t-1} | Z_{1:t-1}, U_{1:t-1}) dw_{t-1} \quad (3.14)$$

In the discrete domain, as there are two possible states of the door FS or OB, so that the transition probability of the state w_t can be presented by.

$$P_c(w_t) = \sum_{j=FS,OB} P(w_t | w_{t-1}^j, u_t) P(w_{t-1}^j | Z_{1:t-1}, U_{1:t-1}) \quad (3.15)$$

The equation (3.15) allows the prediction probability $P_c(w_t)$ to be updated through the transition probability $P(w_t | w_{t-1}, u_t)$ which presents relative reliability of the autonomous navigation algorithm used by the SAM system to pass the door with the state w_t .

Next, the measurement update step calculates the posterior probability $P(w_t | Z_{1:t}, U_{1:t})$ of the state w_t by multiplying the probability $P_c(w_t)$ by the probability $P(z_t | w_t)$ that the measurement z_t may have been observed with assumption that the state w_t is complete.

$$\begin{aligned} P_e(w_t) &= P(w_t | Z_{1:t}, U_{1:t}) \\ &= \eta P(z_t | w_t, Z_{1:t-1}, U_{1:t}) P_c(w_t) \\ &= \eta P(z_t | w_t) P_c(w_t) \end{aligned} \quad (3.16)$$

The probability $P(z_t|w_t)$ that is included to estimate the state w_t of the considering door suggests that the sensor is relatively reliable in measuring the space, and that the uncertain measurement is taken into account in estimating the free space.

The constant η is found so that the distribution of state w_t over its space at time t is normalized

$$\eta = \frac{1}{\sum_{j=FS,OB} P(Z_t | w_t^j, Z_{1:t-1}, U_{1:t}) P(w_t^j | Z_{1:t-1}, U_{1:t})} \quad (3.17)$$

The program structure to implement the Bayesian recursive algorithm is shown as figure 3.20. After initializing the parameters, the prior probability is updated by multiplying the probability of new control $P(w_t|Z_{1:t-1}, U_{1:t})$ to find the prediction probability $P_c(w_t)$ following the equation (3.15). Then, the posterior probability is calculated by the equation (3.16). Finally, the SAM wheelchair makes its own decision about the free space based on the posterior probability $P_e(w_t)$.

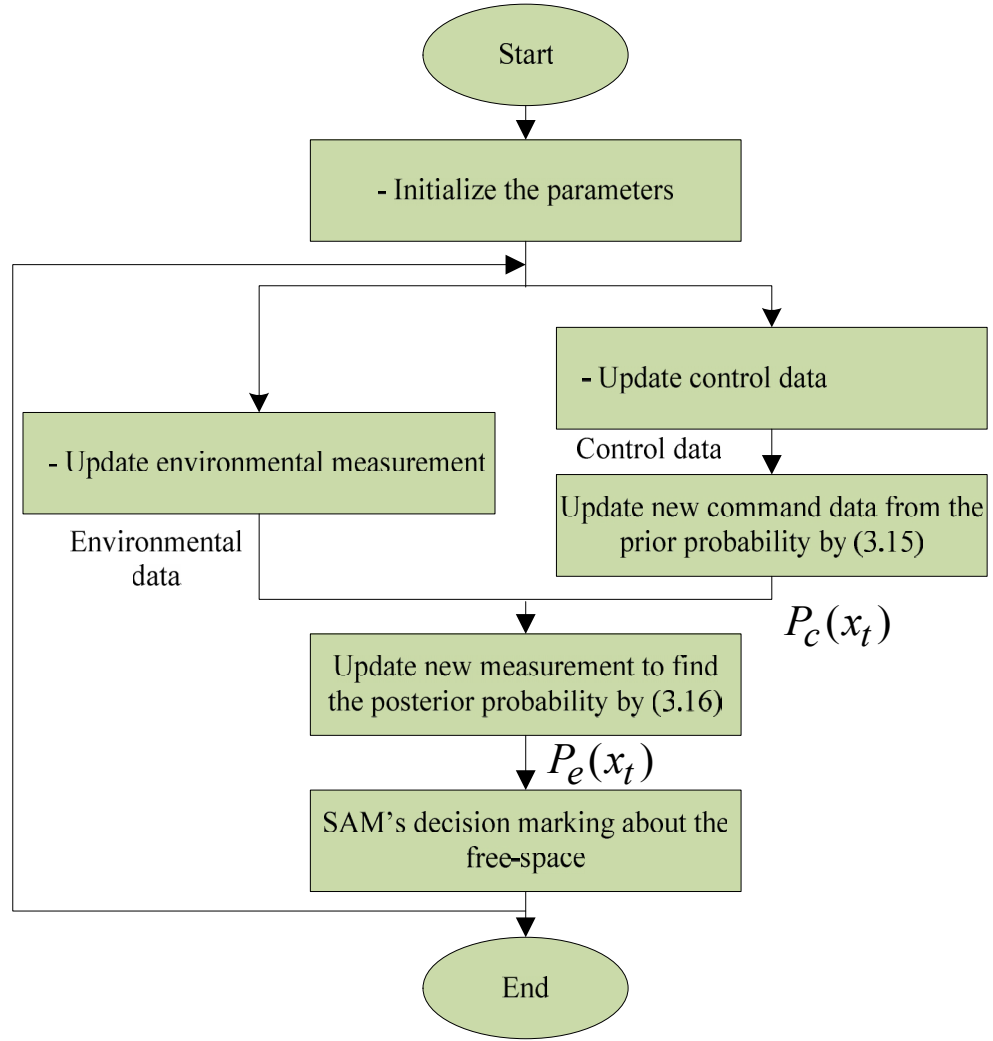


Figure 3.20: The shared control framework using Bayesian Recursive algorithm

3.4 Experimental Results

The purpose of the following real-time experiments is to demonstrate that the system using the Bayesian Recursive technique is able to find appropriate free spaces for the SAM wheelchair to navigate. In the first experiment, the system used the developed technique to autonomously pass through the narrow door. The second experiment repeated the first one except for the fact that there was a moving person in the test course. The experiment showed that the SAM wheelchair was able to stop when the person blocked the door, and continued to pass the door when the path was clear.

3.4.1 Initialization

In order to implement the Bayesian recursive estimation algorithm, some parameters are initialized at the beginning. A considering door is only one of two possible states, free space (FS) or obstacle (OB), and the system does not know the state of the door initially, thus the probabilities about the state w_t of the considering door are assumed to be set equally as follows.

$$\begin{cases} P(w_0 = FS) = 0.5 \\ P(w_0 = OB) = 0.5 \end{cases} \quad (3.18)$$

3.4.1.1 URG-04LX Noise Measurement Modelling

The SAM wheelchair uses the laser range finder URG-04LX to provide the information about a momentary state of the environment in the range scans. The measurements are usually noisy projections of the state due to the limitations of the sensor itself. In particular, the inaccuracy of the size of the door d_m obtained from the URG-04LX comes from three factors: calculation, inaccuracy in range measures, and angular resolution limitation. The common error associated with the measured door width d_m might be estimated based on the sensor's specifications by the following formula.

$$\Delta d = 2(L \times n \times \Delta \theta + \Delta R) \quad (3.19)$$

where: L is the distance of the opening to the sensor

$n \times \Delta \alpha$ is the step scan

ΔR is the range measure accuracy.

Let's take an average value with a assumption that the considering door is 1(m) away from the vehicle, the step scan is $3 \times 0.36^\circ$ and $\Delta R = \pm 10$ (mm). Then, the inaccuracy of the measurement d_m by the equation (3.19) is approximately equal to $\Delta d \approx 0.078$ (m)

From the above calculation, the probability of the measurement of the path opening obtained from the URG-04LX can be assumed to follow the normal distribution, in other words,

$$f(d) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(d-d_m)^2}{2\sigma^2}} \quad (3.20)$$

where: d_m is the measure value determined from the sensor's measure data, and also the mean of the distribution

σ is the standard deviation of the distribution, and chosen to be equal to $\Delta d \approx 0.078$ (m)

3.4.1.2 Free space Passing Capability Modelling

The capability of passing a door of the autonomous navigation system, especially narrow doors, mainly depends on the autonomous navigation algorithm. For simplicity, the vector field histogram is used as the autonomous mobile strategy of the SAM wheelchair for the two following experiments. This algorithm tends to move the vehicle into the middle of the door if it determines that the path is a free space. In order to measure the reliability of the mobile strategy, the SAM wheelchair using the strategy was tested to pass through a door with the width varying from 0.7(m) to 1.2(m). 10 trials were managed for each width of the door, and a number of passing was recorded as the table 3.1.

An assumption that the transition probability with the system performance has a logarithmic form, a logarithmic regression was used to estimate the transition probability $P(w_t|w_{t-1}, u_t)$ which takes a form

$$G(d) = \frac{1}{1 + e^{-\alpha(d-\beta)}} \quad (3.21)$$

Based on the recorded data from the table 3.1, the modelling function can be found

Door size d(m)	Number of passing (10 trails)
0.7	0
0.8	1
0.9	5
1	9
1.1	10
1.2	10

Table 3.1: The performance of passing a door of the SAM wheelchair

$$G(d) = \frac{1}{1 + e^{-27(d-0.9)}} \quad (3.22)$$

The figure 3.21 presents the modelling transition probability of the SAM system using defined autonomous navigation algorithm and the experimental passing opening capability data.

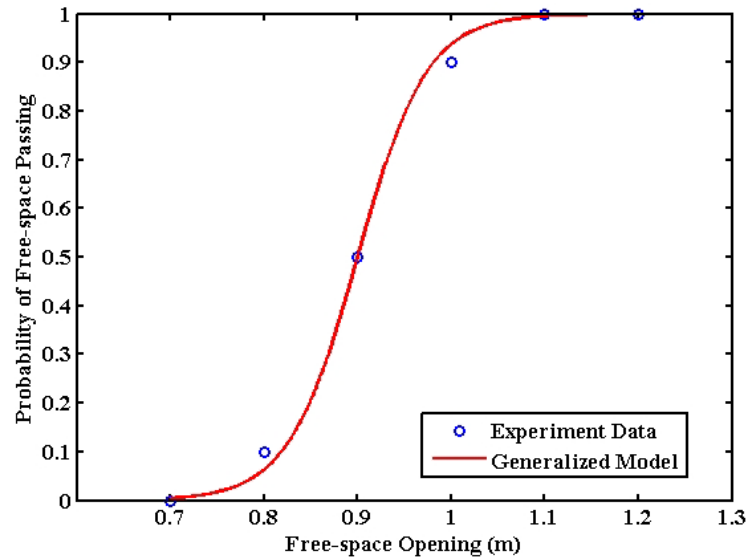


Figure 3.21: The generalized control probability function and experimental data

3.4.2 Experiment 1: Passing a narrow door

After initializing, the SAM wheelchair with the Bayesian recursive method was set about 1.5m away from the door with 1m width as the figure 3.22. At first, the vehicle took on a control action $u_1 = \text{Stop}$, and began to update the prediction probability by the equation (3.15) of the two possible state $w_1 = \text{FS}$ and OB of the door.



Figure 3.22: SAM passing through a narrow free space

$$P_c(w_1 = \text{FS}) = \sum_{j=\text{FS}, \text{OB}} P(w_1 = \text{FS} | w_0^j, u_1 = \text{stop}) P(w_0^j) \quad (3.23)$$

$$P_c(w_1 = \text{OB}) = \sum_{j=\text{FS}, \text{OB}} P(w_1 = \text{OB} | w_0^j, u_1 = \text{stop}) P(w_0^j) \quad (3.24)$$

As the wheelchair took no action, the state w_1 was certainly the same as the state w_0 . That means that $P(w_1 = \text{FS} | w_0 = \text{FS}, u_1 = \text{stop}) = P(w_1 = \text{OB} | w_0 = \text{OB}, u_1 = \text{stop}) = 1$ and $P(w_1 = \text{OB} | w_0 = \text{FS}, u_1 = \text{stop}) = P(w_1 = \text{FS} | w_0 = \text{OB}, u_1 = \text{stop}) = 0$. Replace those values into the equations (3.23) & (3.24), the control probabilities can be obtained as follows

$$\begin{aligned} P_c(w_1 = \text{FS}) &= P(w_1 = \text{FS} | w_0 = \text{FS}, u_1 = \text{stop}) P(w_0 = \text{FS}) \\ &\quad + P(w_1 = \text{FS} | w_0 = \text{OB}, u_1 = \text{stop}) P(w_0 = \text{OB}) \\ &= 1 \times 0.5 + 0 \times 0.5 = 0.5 \end{aligned} \quad (3.25)$$

$$\begin{aligned}
 P_c(w_1 = OB) &= P(w_1 = OB | w_0 = FS, u_1 = stop) P(w_0 = FS) \\
 &\quad + P(w_1 = OB | w_0 = OB, u_1 = stop) P(w_0 = OB) \\
 &= 0 \times 0.5 + 1 \times 0.5 = 0.5
 \end{aligned} \tag{3.26}$$

Next, the SAM wheelchair attained the first measurement about the door which was approximately equal to 0.98 (m). And the posterior probabilities of the state w_1 could be estimated by the equation (3.16) & (3.17)

$$P_e(w_1) = \frac{P(z_1 | w_1) P_c(w_1)}{\sum_{j=FS,OB} P(z_1 | w_1^j) P_c(w_1^j)} \tag{3.27}$$

As the measurement $d_1 = 0.98(m)$ was greater than the safe diameter $D_w = 0.95 (m)$ of the wheelchair, the measurement probabilities could be determined from the normal distribution (3.20) as

$$\begin{aligned}
 P(z_1 = FS | w_1 = FS) &= f(d > D_w = 0.95, d_m = 0.98, \sigma = 0.08) \\
 &= 0.6462
 \end{aligned} \tag{3.28}$$

$$\begin{aligned}
 P(z_1 = FS | w_1 = OB) &= f(d < D_w = 0.95, d_m = 0.98, \sigma = 0.08) \\
 &= 0.3538
 \end{aligned} \tag{3.29}$$

Therefore,

$$\begin{aligned}
 P_e(w_1 = FS) &= \frac{P(z_1 = FS | w_1 = FS) P_c(w_1 = FS)}{P(z_1 = FS | w_1 = FS) P_c(w_1 = FS) + P(z_1 = FS | w_1 = OB) P_c(w_1 = OB)} \\
 &= \frac{0.6462 \times 0.5}{0.6462 \times 0.5 + 0.3538 \times 0.5} = 0.6462
 \end{aligned} \tag{3.30}$$

$$\begin{aligned}
 P_e(w_1 = OB) &= 1 - P_e(w_1 = FS) \\
 &= 0.3538
 \end{aligned} \tag{3.31}$$

The SAM wheelchair makes its own decision to move if the value $P_e(w_t=FS)$ is greater than $P_e(w_t=OB)$. If it does not meet that condition, then the system will stop. From the values from (3.30) & (3.31), the wheelchair generated $u_2 = \text{move}$, and started to move towards the door. The next update cycle was calculated as follows:

$$P_c(w_2 = FS) = \sum_{j=FS,OB} P(w_2 = FS | w_1^j, u_2 = move) P(w_1^j) \quad (3.32)$$

$$P_c(w_2 = OB) = \sum_{j=FS,OB} P(w_2 = OB | w_1^j, u_2 = move) P(w_1^j) \quad (3.33)$$

The state transition probabilities would be found from the modelling control probability function (3.22) with the measurement of the door $d_1 = 0.98$ (m) from the previous step.

$$\begin{aligned} P(w_2 = FS | w_1 = FS, u_2 = move) &= G_{(d=d_1)} \\ &= \frac{1}{1 + e^{-27(0.98-0.9)}} = 0.8966 \end{aligned} \quad (3.34)$$

$$\begin{aligned} P(w_2 = OB | w_1 = FS, u_2 = move) &= 1 - G_{(d=d_1)} \\ &= 1 - 0.8966 = 0.1034 \end{aligned} \quad (3.35)$$

Hence,

$$\begin{aligned} P_c(w_2 = FS) &= P(w_2 = FS | w_1 = FS, u_2 = move) P(w_1 = FS) \\ &\quad + P(w_2 = FS | w_1 = OB, u_2 = move) P(w_1 = OB) \\ &= 0.8966 \times 0.6462 + 0 \times 0.3538 = 0.5794 \end{aligned} \quad (3.36)$$

$$\begin{aligned} P_c(w_2 = OB) &= P(w_2 = OB | w_1 = FS, u_2 = move) P(w_1 = FS) \\ &\quad + P(w_2 = OB | w_1 = OB, u_2 = move) P(w_1 = OB) \\ &= 0.1034 \times 0.6462 + 1 \times 0.3538 = 0.4206 \end{aligned} \quad (3.37)$$

The measurement value d_2 of this step is 1.02 (m), the posterior probabilities of the state w_2 is calculated similar to the previous step.

$$\begin{aligned} P(z_2 = FS | w_2 = FS) &= f(d > D_w = 0.95, d_m = 1.02, \sigma = 0.08) \\ &= 0.8092 \end{aligned} \quad (3.38)$$

$$\begin{aligned} P(z_2 = FS | w_2 = OB) &= f(d < D_w = 0.95, d_m = 1.02, \sigma = 0.08) \\ &= 0.1908 \end{aligned} \quad (3.39)$$

$$P_e(w_2 = FS) = \frac{P(z_2 = FS | w_2 = FS) P_c(w_2 = FS)}{P(z_2 = FS | w_2 = FS) P_c(w_2 = FS) + P(z_2 = FS | w_2 = OB) P_c(w_2 = OB)} \quad (3.40)$$

$$= \frac{0.8106 \times 0.581}{0.8106 \times 0.581 + 0.1894 \times 0.352} = 0.8539$$

$$P_e(w_2 = OB) = 1 - P_e(w_2 = FS) = 0.1461 \quad (3.41)$$

Similarly, the SAM wheelchair continued to move towards the door because its belief about the door with state $w_2=FS$ was improved. The figure 3.18 shows the posterior probabilities of the states ($w_t=FS$ and $w_t = OB$) of the door in a few next time cycles. Although the measurement values of the opening vary in from 0.97 (m) to 1.06 (m), the system

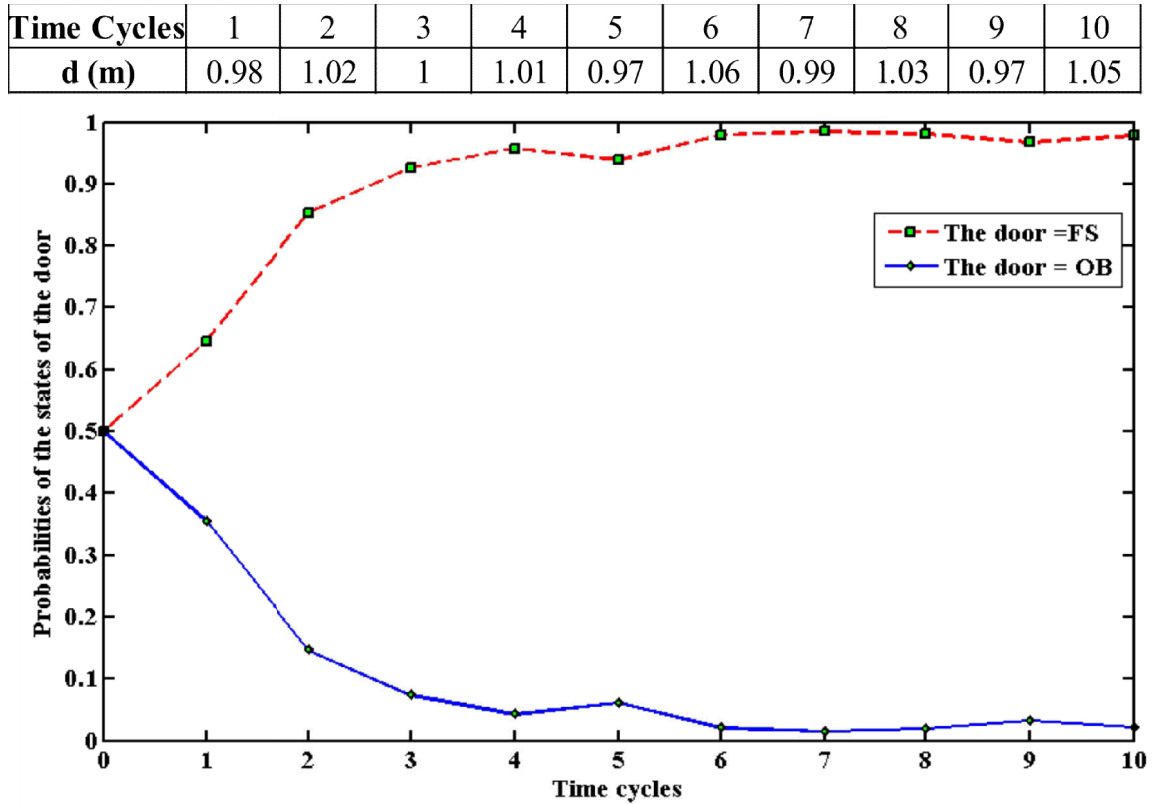


Figure 3.23: The states of the door are recursively updated under the Bayesian Framework

consistently believed that the door was a free space as the probability $P_e(w_t=FS)$ increased and saturated at 0.9674 just after a few time cycles while the probability $P_e(w_t=OB)$ reached to 0.0326

3.4.3 Experiment 2: Passing a narrow door with a moving person

In this experiment, the SAM wheelchair was still tested to pass through the door with 1m width. However, while navigating, a person passed through the door as shown in the figure 3.19. As the wheelchair was approaching the door with the belief that the door was a free space corresponding to the $P_e(w_t=FS) = 0.9644$, and $P_e(w_t=OB) = 0.0356$ at the time cycle $t = 241$, a person moved in the door and blocked the way of the wheelchair in a few seconds. The system detected the change of the environment and updated the state probabilities in the real-time similar to the first experiment.



Figure 3.24: SAM passing through a narrow free space with a moving person

Figure 3.24 shows the result of the state probabilities ($w_t = FS$ and $w_t = OB$) of the door when a person occupied the door. As the person went into the door, the opening of the door was smaller, leading to the probability $w_t = FS$ of the door decreased below that of $w_t = OB$. Therefore, the system automatically stopped just after 1 time cycle as the person moved into its navigating field. The belief that the door was a free space stayed very low until the user moved out of the door. The $P(w_t=FS)$ resumed the higher values as the door was clear after 1 time cycle.

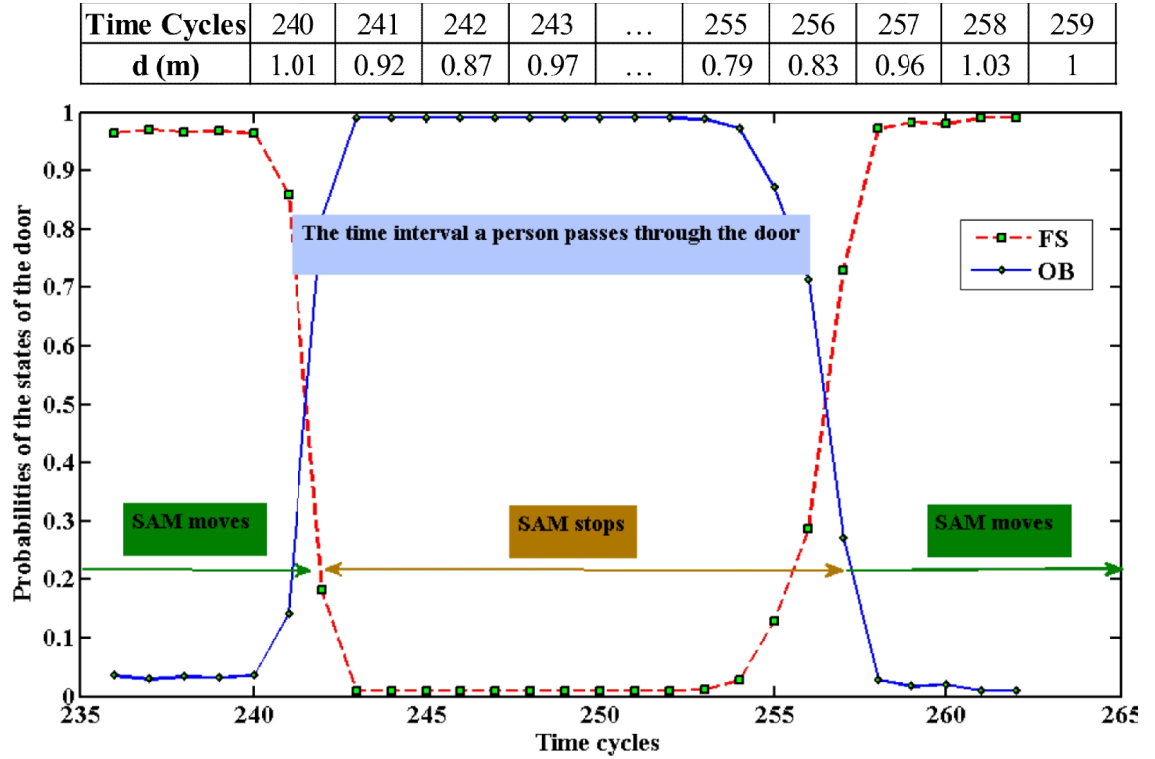


Figure 3.25: The states of the door are recursively updated with a moving person

3.5 Discussion

Briefly, there are two main parts constituting this chapter. In the first part of this chapter, we introduced the design of our intelligent wheelchair system. In particular, we gave an overview of the different components assembled into the SAM system. For the second part of this chapter, we proposed an advanced method to construct a real-time map and Bayesian recursive technique in order to find the free spaces for navigation corresponding to the available environmental information obtained from the sensor system.

As compared to other works exposed previously, our design is intended to facilitate and ease the building process of the vehicle in which any commercial underlying wheelchair platform can easily turn into an intelligent system without heavy modification. In particular, our platform in this thesis is based on a Dynamic Controls powered wheelchair. All

additional items are attached on the back or on the side of the chair without affecting the hardware of the wheelchair. Perhaps, the hardest task in the design only reprograms the parameters of the master remote control to integrate our virtual joystick model into the system through which the computer can take the control. Nevertheless, it can be completely done by software. This means that a large number of standardized powered wheelchairs can be changed with the same software facilitating commercializing the design.

A real-time map for the autonomous navigation was also developed in this chapter. The basis of the map building is to combine the data of the laser range finder sensor and wheel encoders to create an image of the obstacle distribution around possible navigation directions of the wheelchair. The laser range finder plays a major role in collecting local obstacle position information while encoder data assist to infer obstacles that might cause collisions but are out of range of the laser sensor. The system asks these sensors to sample the local environment in each 100ms (sampling frequency is 10Hz). By doing so, the map data always updates in real-time providing the reliability for the autonomous navigation algorithm developed in the next chapter.

The other main part of this chapter is to determine free spaces to navigate for the system. Unlike many other works that rely heavily on the vehicle point assumption costing many computing steps, the technique used here reduces those steps by simply comparing the sizes of considering spaces to the wheelchair width to find the possible candidates. The free spaces are, then, found as the result of the combination of the measurement from the sensor system and the capability of passing through the spaces corresponding to its navigation strategy. The algorithm is recursively applied to detect the change of the environment and find free spaces to navigate. The experimental results showed that the method is able to determine the appropriate free spaces in the real - time. Because of taking noises into account, this technique lessens the effect of the hardware device limitation in the calculation. Moreover, as expressing the free space results in terms of the probabilities, the SAM wheelchair can make the most optimal decisions about the free spaces based on the probabilistic reasoning.

As the surrounding free – spaces are found, it is now necessary to obtain an autonomous algorithm to assist the system to navigate. In the next chapter, we will focus on using Bayesian neural network to create an effectively autonomous navigation method that not only safely avoids obstacles in real-time, but also produces smooth and optimal trajectories in the unknown environment.

Chapter 4

Autonomous Navigation Using Bayesian Neural Networks

4.1 Introduction

In chapter 3, the advanced instrumentation of an intelligent wheelchair system is presented in detail. Additionally, the real-time map method was built from the on-board sensory information which allows the system to be capable of navigating in an unknown environment. Finally, a recursive Bayesian technique is applied to find a number of free spaces for the SAM intelligent wheelchair.

This chapter will mainly focus on developing a robust autonomous mobile algorithm which assists SAM to effectively navigate into the free spaces which have been determined. Obviously, it is necessary to provide the mobile support to the impaired users. Due to either physical or mental limitations, many users with disabilities find it very difficult or impossible to control a wheelchair at a low level of navigation, such as guiding the wheelchair while avoiding obstacles in a dense environment, or entering a narrow door without collision, etc. The clinical findings (Fehr, Langbein & Skaar 2000; Wan LM 2010) provide insight into the importance of devising a computer – controlled platform to assist the users by reducing their workload and increasing safety.

Despite the fact that various autonomous navigation techniques have been applied in the intelligent wheelchair field, most of them are hardly able to satisfy several important

quality requirements such as smoothness, comfort or stability. This is because the methods were originally developed for autonomous mobile robots rather than assistive vehicles for people with disabilities. In order to overcome the drawbacks, this chapter will present an advanced non-collision autonomous navigation for the SAM wheelchair that not only is able to successfully navigate in the unknown environment, but also satisfies the requirements of an assistive system. This would include producing smooth trajectories, as well as creating the comfort for the user during daily driving manoeuvres.

To create an effectively autonomous navigation controller with many criteria required for the assistive system, a Bayesian neural network is proposed to achieve the goal. The principle of this technique relies on the ability of an artificial neural network to learn how to react in a number of certain obstacle avoidance situations. Through a number of patterns during the training process under the Bayesian supervision, the optimal trained network would obtain necessary skills to drive the system with predetermined characteristics. In other words, obstacle avoidance skills and different quality requirements for the system can also be imparted to the network by deliberately managing the patterns which is normally impossible for other techniques.

This chapter is organized as follows. An autonomous navigation system will be first introduced for the SAM wheelchair. Then, a practical Bayesian framework will be developed to estimate the most proper network structure as well as its weight values on the basis of training data. Finally, this chapter will show real-time experiments to evaluate the performance of the designed obstacle avoidance method.

4.2 Autonomous Navigation System Based on a Bayesian Neural Network

The autonomous control architecture for SAM is depicted in the figure 4.1. SAM uses its sensors to perceive the surrounding environment. From the obtained data, the procedures discussed in chapter 3 produce the real-time map as well as the free spaces to navigate in the local environment. The SAM wheelchair is designed on the basis of a standard

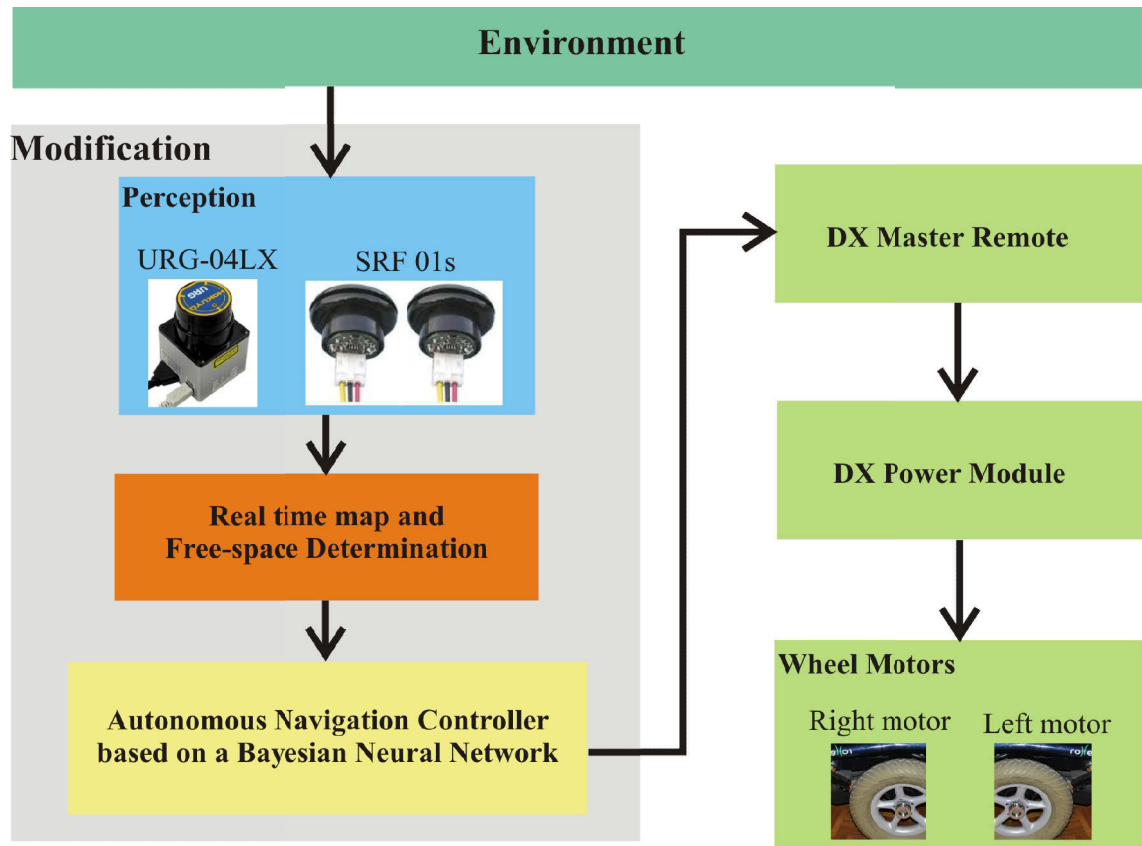


Figure 4.1: Autonomous Control Function Diagram for SAM wheelchair

commercial wheelchair, therefore, the control signals generated by the autonomous navigation controller would be transferred to the DX master remote via our design DX module – called Virtual joystick module. Finally, based on the received values, the DX master would tell the DX power module to control the wheel motors accordingly.

The autonomous navigation controller is designed in order to safely manoeuvre the system into the intended free space given the perceived environment data. In other words, the inputs of the controller would be the data obtained from the sensors, and the outputs of the controller would be the values to control the wheel motors. The problem which needs to be solved here is that the controller with a set of the environment data should drive the wheelchair to not only avoid surrounding obstacles but also produce the smooth trajectories in order to comfort the user sitting on the platform. Therefore, typical methods for mobile robots such as VFF (Petry et al. 2010), VHF (Lopes, Nunes & Vaz 2010; Tang et al. 2012) or Dynamic window (Yokozuka et al. 2012) are hardly suitable to apply to the assistive

system like the intelligent wheelchair application. Instead, the artificial Bayesian neural network becomes an attractive solution for the SAM system.

The idea behind this technique is simple: required manoeuvring skills, which include driving the wheelchair in harmony with the surrounding environment, optimal steering around the obstacles, or producing smooth trajectories, etc, would deliberately result in patterns and use them to train a neural network. During the training process, Bayesian theory is applied in order to choose the most optimal structure and the most optimal weight values for that particular data. By doing so, the framework will prevent the network from over - fitting during training and improve the generalization of the resultant network.

4.2.1 Wheelchair Control Signals

Originally, the standard powered wheel motors were only controlled by joystick signals. In order to be controlled by signals generated from the on-board computer, an additional DX module, called Virtual joystick module, has been designed for that purpose. The function of this module is to simply receive signals within the pre-defined range, then transfer those to the DX master remote via DX bus. The figure 4.2 shows the connection diagram of the module to the system, and the functional pins of the module.

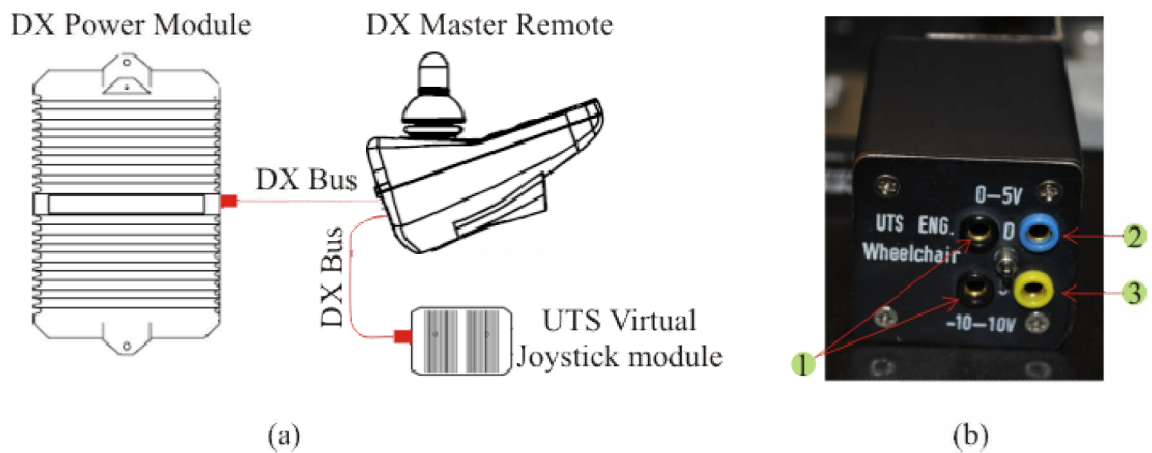


Figure 4.2: UTS Virtual Joystick Module
a) Connection of the module to the system; b) The module's inputs: 1) Ground, 2) Steering signal, 3) Speed Signal.

As its name implies, the signals fed to this module, identical to the joystick's signals, include two signals sharing the common ground input pin 1. The signal connected to the pin 2 is used to control the translational speed of the wheelchair, while the pin 3's signal is to steer the vehicle. The signals' range is depicted in the figure 4.3

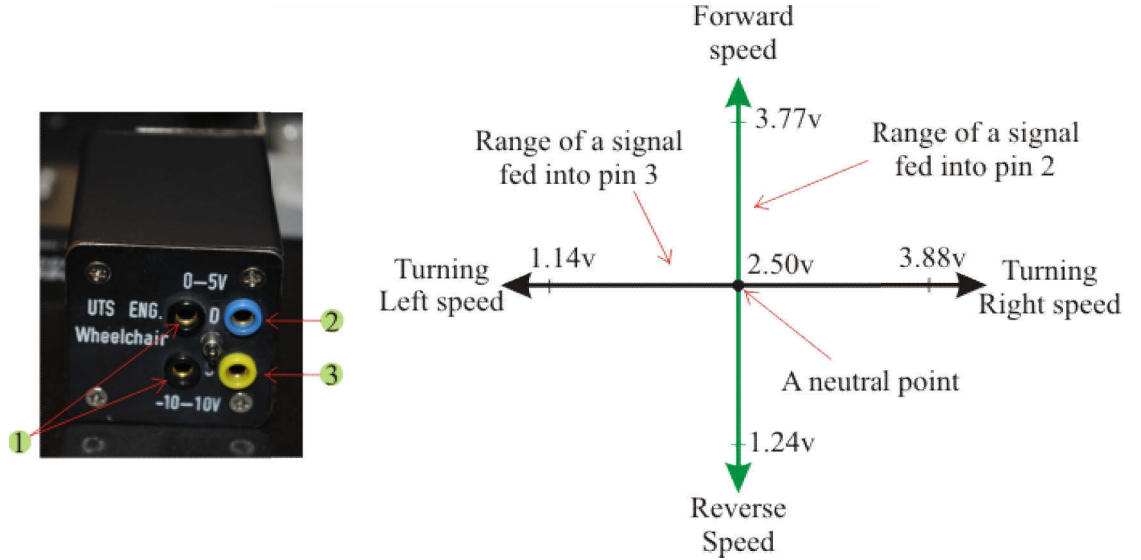


Figure 4.3: The UTS Virtual Joystick Module inputs and range of the signals
1: Ground input, 2: Steering signal input, 3 Speed signal input

The translational speed signal is about $2.5V \pm 1.27V$, the $2.5V$ value corresponds to the speed equal to zero, while the forward fastest speed ($\approx 1m/s$) is reached with $3.77V$, and the reverse fastest speed is with $1.24V$. Similarly, the rotational signal range is about $2.5V \pm 1.38V$. Values greater than $2.5V$ correspond to the turning right state of the wheelchair. On the contrary, the values less than $2.5V$ correspond to the turning left state of the wheelchair.

When designing the obstacle avoidance neural network for the system, it is convenient to refer to terms like the translational velocity and rotational velocity. In practice, as these values are obtained, it is necessary to convert them into the signals' values written into the UTS virtual joystick module. This can be done by just a linearly rescaling approach. Assuming that the range of the translational speed v of the vehicle is 0 to $1m/s$, then when $v = 0$, the signal value y sent to the pin 2 of the module is expected to be $2.5V$, and if v

=1m/s, y should be 3.77V. Therefore, the relationship between y and v can be determined by

$$y = 2.5 + 1.27v \quad (v) \quad (4.1)$$

Similarly, the signal x fed into the pin 3 of the module might be calculated based on the value of the rotational velocity or turning speed ϕ in radian per second by the equation

$$x = 2.5 + 1.3 \frac{2\phi}{\pi} \quad (v) \quad (4.2)$$

Inversely, the signal x and y are sometimes acquired, and the velocity v and turning speed ϕ are needed to be expressed by

$$v = \frac{1}{1.27}(y - 2.5) \quad (m / s) \quad (4.3)$$

$$\phi = \frac{\pi}{2.6}(x - 2.5) \quad (rad) \quad (4.4)$$

In the following section, the artificial neural network architecture for the autonomous navigation controller will be discussed. It would be seen later that the equations (4.1) and (4.2) are used to convert the network's outputs into real control signal values. The equations (4.3) and (4.4) are for the training data collection in which the real control signals must be reverted to the desired outputs of the training data.

4.2.2 Artificial Neural Network for Autonomous Navigation

In recent years, neural networks have shown great promise in identifying complex nonlinear mappings from observed data, and have many applications in non-linear control. Thus, artificial neural networks are believed to be well suited for learning the complex internal mapping from sensory inputs to control action outputs. In this application, a feed forward neural network architecture is employed with one hidden layer for modelling the human skills utilized in navigating an obstacle course. This learning architecture is selected because of its representation capability and its practical implementation.

The decision of the neural network architecture selection is very critical for this real-time application. The chosen structure should be complex enough to represent the non-linear input-output relationship, but also be simple enough in order to produce fast response in the real-time. The neural networks with a single layer are generally capable of representing only linearly separable functions or linearly separable decision domains, and thus, they are obviously unsuitable for the SAM wheelchair application. It is widely acknowledged that one hidden layered network might approximate arbitrarily well any functional continuous mapping from one finite-dimensional space to another (Bishop 1995; Mark 1999; Sarles 2002). In addition a network should be, in practice, designed to have as few layers as possible because the training process takes dramatically longer and it is much harder to find the global minima during training when more hidden layers are used (Bishop 1995). Based on the argument, one hidden layer would be the first choice for any practical feed-forward network design.

Although a feed-forward network with one hidden layer, in principle, could be used to map the raw input data directly onto the required final output values, it is always advantageous to apply pre-processing transformations to the input data it is presenting to a network.

4.2.2.1 Pre-Processing Environmental Data

The real-time map comprises 180 points associated with the distances of all surrounding obstacles. All the data can be directly put into the inputs of the network, however, from a practical point of view, it is necessary to transform the data into new representation to increase the generalization as well as shorten the computational time. In an effort to improve the generalization, the pre-processing used in our report (V.Nguyen. et al. 2012) simply reduced the dimensionality of the input data by removing a part of redundant data. The procedure is taken by dividing the whole data into a number of clusters, and in each cluster, the closest point to the wheelchair is selected as an input of the network. For instance, the laser's 180 data points, corresponding to the distance of the obstacles at angles from 0 to 180 degrees to the wheelchair reference point, are divided into 36 clusters. Thus,

there are five points with respect to five consecutive angles in each cluster. The minimum data point in each 5 degrees is chosen as the input value.

$$I_i = \min\{r_{5i+j}\} \text{ for } j = 1 \text{ to } 5 \quad (4.5)$$

Further improvement on pre-processing the environment information is applied in the report (V.Nguyen. et al. 2013). Rather than using raw data, the data is transformed by considering how surrounding obstacles affect the navigating process to extract relevant information before it is presented to the network. Experiment results show that the new proposed approach increases the generalization and efficiency of the trained network which helps the wheelchair not only effectively avoid obstacles while navigating to the local target, but also considerably enhances the door passing performance of the system.

Relevant information of the surrounding environment can be found by taking into consideration the wheelchair's control in a relationship of local obstacles and the navigation target T. The figure 4.4 a) displays all instantaneously measured obstacle distances to the wheelchair in a polar form at a time, and wheelchair dimensions. It is obvious that not all of these obstacles play an equally important role in navigating the wheelchair. In particular, obstacles in the A and B regions have significant effects on both speed and steering direction of the wheelchair while others far from the wheelchair play a less meaningful role. Thus, it is necessary to select important environment features and represent them in appropriate forms in order to improve the performance of the trained neural network.

The following steps represent how to transform the real - time environment map to obtain a number of inputs for a neural network as displayed in the figure 4.4 b, c, and d.

Step 1

Remove unnecessary environment data points to reduce the dimensionality of the input network. For instance, divide the 180 URG-04LX data points into 36 sections - 5 points per each section, and the closest obstacle point of each section represents that section. By the number of tries, 5 - point section is likely to be optimal because if the number of points in each section is chosen to be too large, some important points might be lost. If the number is

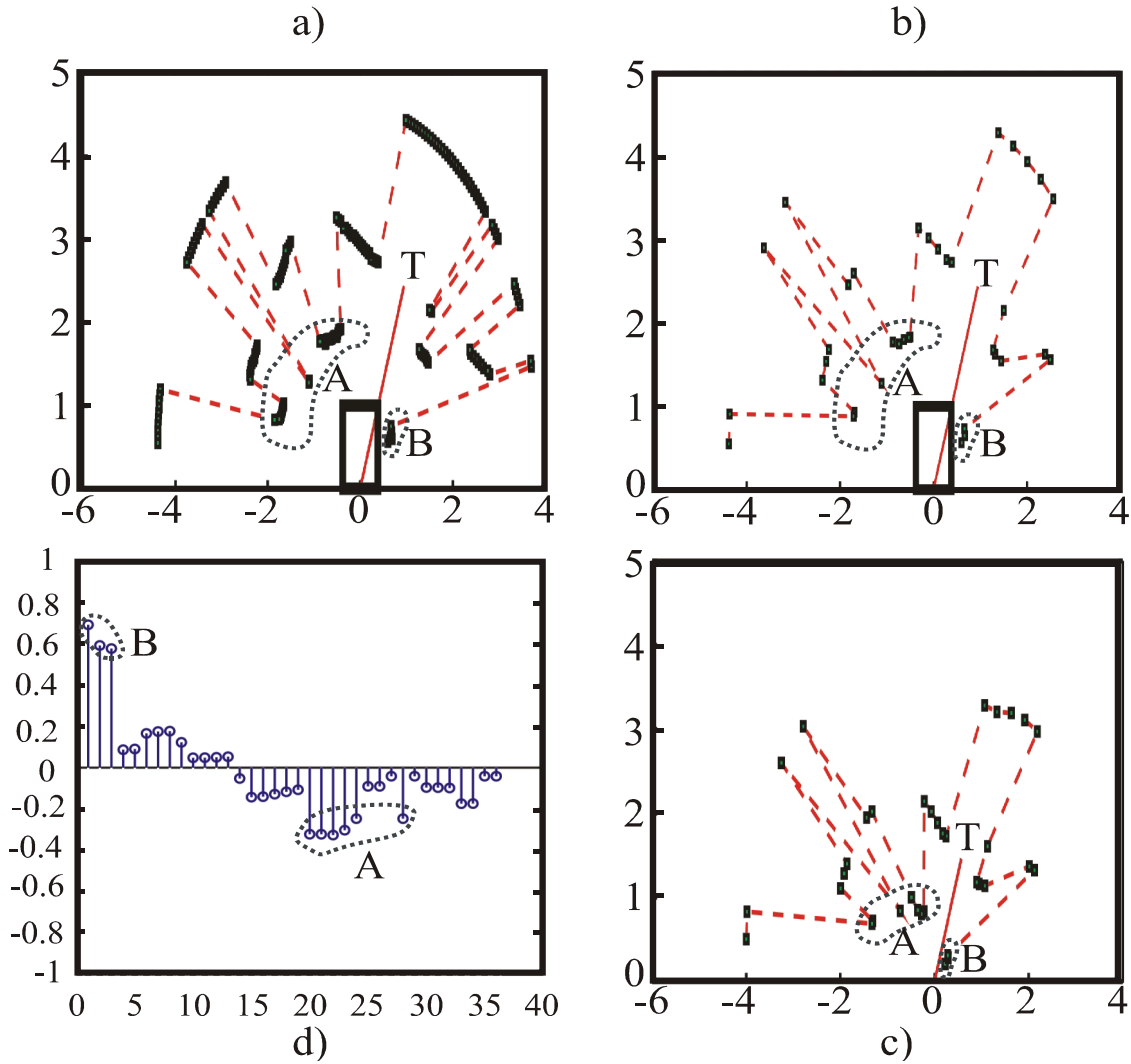


Figure 4.4: Steps to transform URG-04LX data

too small, it results in too many sections. The result of this step is displayed in the figure 4.4 b).

Step 2

From chosen data from step 1, take the wheelchair's dimensions into account to find the real distance of selected obstacles to the wheelchair shown in the figure 4.4 c). This leads to the equality of obstacle distance values for all directions.

Step 3

Transform all values from step 2 by this normalized formula

$$I_{nth} = \frac{a}{(d_{nth} + b)^2} \quad (4.6)$$

where: I_{nth} and d_{nth} are values of n^{th} section after and before transforming. a , b are normalization constants that are determined such as I_{nth} changes from 0 to 1 when d_{nth} ranges 4(m) to 0.1(m).

Step 4

As effects of obstacles in varying positions to the target are different, a positive or negative sign is assigned for each value of section. For example, those lying on the right side of the travel direction tend to push the vehicle towards the left, resulting in the steering angle being larger than the angle to the target, so that they have positive effects. Similarly, the left side obstacles lead the steering angle to be smaller than the goal angle, and therefore have negative impacts.

The results of step 3 and 4 are represented in the figure 4.4 d).

4.2.2.2 Feed Forward Neural Network Architecture

The network is designed on the basis of the inputs of processed environment data as well as a direction of travel and the outputs of the control values to manoeuvre SAM without collisions. In other words, the inputs of the network would be comprised of 36 values as a result of pre-processing the URG-04LX data in the previous section and one value that is the direction of travel, while its outputs are two control values corresponding to the signals to control the translational velocity and rotational velocity of the wheel motors.

The figure 4.5 shows a feed-forward neural network obstacle avoidance configured to provide a framework for representing non-linear functional mappings between a set of input variables and a set of output variables. In this network, there are 37 inputs, M hidden units, and 2 output units. The output of the j^{th} hidden unit is determined by forming a weighted linear combination of the 37 input values, and adding a bias by the following equation.

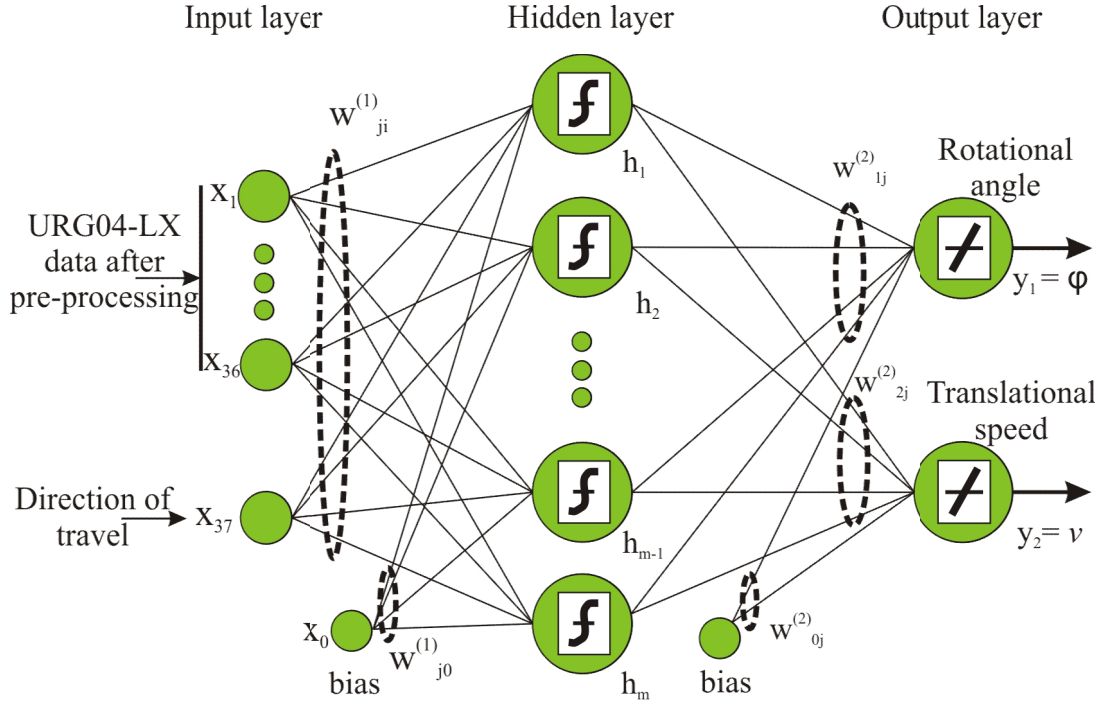


Figure 4.5: Feed-forward neural network structure for obstacle avoidance

$$a_j = \sum_{i=1}^{37} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \quad (4.7)$$

where $w_{ji}^{(1)}$ is the weight from the input i^{th} to the hidden unit j^{th} in the first layer and $w_{j0}^{(1)}$ is the bias for the hidden unit j^{th} . If we are considering the bias term as being weights from an extra input $x_0 = 1$, then (4.7) can be rearranged to the form

$$a_j = \sum_{i=0}^{37} w_{ji}^{(1)} x_i \quad (4.8)$$

The activation of the hidden unit j^{th} then can be obtained by transforming the linear sum using a sigmoid function $g(x)$:

$$h_j = g(a_j) = \frac{1}{1 + e^{-a_j}} \quad (4.9)$$

This function is chosen because it is easy to differentiate, and thus can dramatically reduce the computation burden for training. The outputs of the network can be determined by transforming the activation of the hidden units using a second layer of processing units. For

each output unit k ($k = 1, 2$), first we find the linear combination of the output of the hidden units,

$$a_k = \sum_{j=1}^M w_{kj}^{(2)} h_j + w_{k0}^{(2)} \quad (4.10)$$

Again we can absorb the bias and rewrite the above equation to

$$a_k = \sum_{j=0}^M w_{kj}^{(2)} h_j \quad (4.11)$$

We use identity function $g_2(x)$ for the output layer, then applying the activation function $g_2(x)$ to the equation (4.11) we can get the k^{th} output

$$y_k = g_2(a_k) = a_k \quad (4.12)$$

Combining equations (4.8), (4.9), (4.11) and (4.12) we receive the complete representation of the network as

$$y_k = \sum_{j=0}^M w_{kj}^{(2)} g\left(\sum_{i=0}^{37} w_{ji}^{(1)} x_i\right) \quad (4.13)$$

The equation (4.13) shows the relationship of the inputs associated with the sensory information, the direction of travel and the network's outputs associated with the translated velocity and rotational speed of the vehicle. This relationship depends on the structure of the network or the number of hidden units M and the network weight values which are determined through the training process.

4.3 Bayesian Neural Network Training

4.3.1 Network learning

The functionality of a neural network is determined by the combination of the topology and the weights of the connections within the network. The supervised training model as

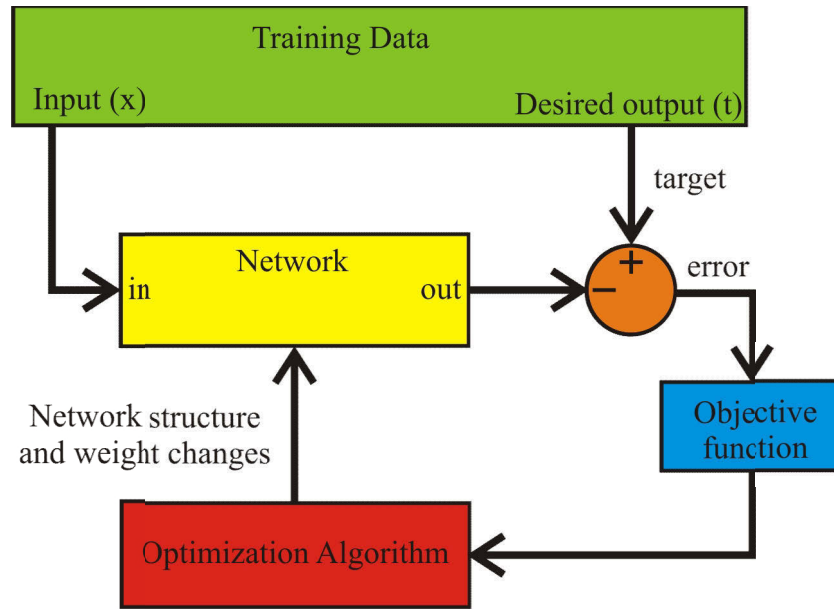


Figure 4.6: Neural network training model for obstacle avoidance task

depicted in the figure 4.6 can be used to adjust the topology and weights to make the network learn the relationship between the inputs and the targets (outputs).

The network is trained by providing it with the training data comprising of inputs x and desired outputs (target value t). These input-output pairs are considered as an ‘external teacher’ described later. The difference between the real outputs produced by the network and the desired outputs is employed by the optimization algorithm to adapt the structure and weights of the network. It is often posed as a function approximation problem - given training data consisting of pairs of input patterns x , and corresponding target t , the goal is to find a function $y = f(x)$ that matches the desired response for each training input.

In the most conventional optimization methods, the network structure is required to hold fixed in order to find the optimum set of weights. And thus, determining the best number of hidden units always involving many factors becomes a very tricky task for this application. In order to overcome the difficulty, a practical Bayesian framework is proposed, which is integrated into the Levenberg – Marquardt network training algorithm in order to find the most optimal structure network for this application. There are two phrases involved in this framework: optimizing weight values and optimizing the structure. In particular, the number of hidden units of the network is allowed to vary during the training process, and

each structure as well as its optimum set of weights are evaluated on the basis of evidence value and compared with other structures. The structure and its weights are believed to be the best optimal ones if the corresponding network has the highest evidence performance.

In the next section, we will begin by discussing the Levenberg-Marquardt optimization algorithm that is employed to optimize a set of weights for each particular network structure.

4.3.2 Levenberg-Marquardt Optimization Algorithm

The Levenberg – Marquardt is an iterative technique that locates a local minimum of a multivariate function that is expressed as the sum of squares of several non-linear, real-valued functions. It has become a standard technique for non-linear least-squares problems, widely adopted in various disciplines for dealing with data-fitting applications. This technique is opted for this application because this is one of the fastest training methods and provides a steady convergence capability for the network training process.

We start by laying out some notation. Let the input vector with N elements be x and its associated correct output response be t . Suppose that the neural network produces an actual response of y , the squared error is then $e = (y-t)^2$. The network weight w is sought to minimize the error function $E(w)$ which is expressed in a quadratic form as follows

$$\begin{aligned} E(w) &= \frac{1}{2} \sum_{n=1}^N \{y(x^n, w) - t^n\}^2 \\ &= \sum_{n=1}^N e^2(w) = e^T(w) e(w) \end{aligned} \quad (4.14)$$

The relationship of the error function between an iteration k and the next iteration $(k+1)$ might be determined based on the second-order Taylor series

$$E(w_{k+1}) = E(w_k + \Delta w_k) \approx E(w_k) + \frac{\partial E(w_k)}{\partial w} \Delta w_k + \frac{1}{2} \Delta w_k^T \frac{\partial^2 E(w_k)}{\partial w^2} \Delta w_k \quad (4.15)$$

A local minimum of the error function can be reached by taking the gradient of the function with respect to Δw_k and setting it equal to zero, we find

$$w_{k+1} = w_k + A_k^{-1} g_k \quad (4.16)$$

where $g_k = \nabla E(w)|_{w=w_k}$, and $A_k = \nabla^2 E(w)|_{w=w_k}$

Both the gradient and the second order of the error function would be obtained through the user of straightforward application of the chain rule and the multiplication rules. First, the gradient would be

$$\nabla E(w) = \frac{\partial E(w)}{\partial w} = 2 \sum_{i=1}^N e(w) \frac{\partial e(w)}{\partial w} \quad (4.17)$$

In the matrix form

$$\nabla E(w) = 2J^T(w)e(w) \quad (4.18)$$

where

$$J(w) = \begin{bmatrix} \frac{\partial e_1(w)}{\partial w_1} & \frac{\partial e_1(w)}{\partial w_2} & \dots & \frac{\partial e_1(w)}{\partial w_n} \\ \frac{\partial e_2(w)}{\partial w_1} & \frac{\partial e_2(w)}{\partial w_2} & \dots & \frac{\partial e_2(w)}{\partial w_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_N(w)}{\partial w_1} & \frac{\partial e_N(w)}{\partial w_2} & \dots & \frac{\partial e_N(w)}{\partial w_n} \end{bmatrix} \text{ is the Jacobian matrix.}$$

Next, the second order of the error function forms the Hessian matrix whose k, j element would be

$$[\nabla^2 E(w)]_{k,j} = \frac{\partial^2 E(w)}{\partial w_k \partial w_j} = 2 \sum_{i=1}^N \left\{ \frac{\partial e(w)}{\partial w_k} \frac{\partial e(w)}{\partial w_j} + e(w) \frac{\partial^2 e(w)}{\partial w_k \partial w_j} \right\} \quad (4.19)$$

In the matrix form

$$A = \nabla^2 F(w) = 2J^T(w)J(w) + 2S(w) \quad (4.20)$$

$$\text{where } S(w) = \sum_{i=1}^N e_i(w) \nabla^2 e_i(w)$$

It is usually reasonable to assume that errors in the training set are fairly independent and identically distributed around a mean of zero. Thus, the term $S(w)$ can be ignored, and the Hessian matrix can be approximated as

$$\nabla^2 F(x) \approx 2J^T(w)J(w) \quad (4.21)$$

Substitute the equations (4.18) and (4.21) into the equation (4.16), we obtain

$$w_{k+1} = w_k - [J^T(w_k)J(w_k)]^{-1} J^T(w_k) e(w_k) \quad (4.22)$$

This update method is known as Gauss-Newton. Its advantage over the standard Newton's method is that it does not require calculation of second derivatives. However, the matrix Hessian $A = J^T J$ might not be invertible. In order to overcome this, an invertible approximate Hessian matrix is introduced

$$G = A + \mu I \quad (4.23)$$

where I is a unit matrix and μ is a scaling factor.

By increasing μ until a large enough number, the matrix G is certainly invertible, and this leads to the Levenberg – Marquardt algorithm

$$w_{k+1} = w_k - [J^T(w_k)J(w_k) + \mu_k I]^{-1} J^T(w_k) e(w_k) \quad (4.24)$$

Or

$$\Delta w_k = -[J^T(w_k)J(w_k) + \mu_k I]^{-1} J^T(w_k) e(w_k) \quad (4.25)$$

This algorithm has a very useful feature that as μ_k is increased, it approaches the steepest descent algorithm, while as μ_k is decreased to zero the algorithm becomes Gauss – Newton. In other words, the Levenberg – Marquardt can be thought of as a combination of steepest

decent and the Gauss – Newton method. When the current solution is far from a local minimum, the algorithm behaves like a steepest descent method: slow, but guaranteed to converge. When the current solution is close to a local minimum, it becomes a Gauss - Newton method and exhibits fast convergence.

4.3.3 Bayesian Framework for Feed-Forward Neural Networks

Maximum likelihood neural network training technique is equivalent to minimization of an error function. However, this framework is typically unable to compare different models in terms of the generalization, so that partitioning of the data set must be implemented to select an appropriate level of complexity through such techniques as cross-validation.

By contrast, Bayesian methods are able to deal with the issue of model selection without the need to use cross-validation. This approach, first introduced by Mackay (Mackay 1992), provides a framework to choose the most probable model corresponding to the specific training data D in automatic fashion. In the Bayesian framework, a number of models with different complexity are considered on the basis of training data and represented the relative degrees of belief in the different values. Initially, each model is assigned to a prior probability. Once the data has been observed, the prior probability can be converted to a posterior probability through the use of Bayes' theorem.

$$p(H_i | D) = \frac{p(D | H_i)p(H_i)}{p(D)} \quad (4.26)$$

Where $p(H_i)$ is a prior probability for the model H_i . If we do not prefer one model over another, we would assign equal priors for all models. As $p(D)$ does not depend on the model H_i , the network model H_i can be compared based on $p(D|H_i)$ which is generally called the evidence for the model H_i . The Bayesian approach illustrates that the evidence favours models neither too simple nor too complex which will typically have the best generalization performance. This indicates that this approach could be used to select a particular model for which the evidence is largest.

4.3.3.1 Prior Weight Distribution

The problem of training a neural network with a given architecture, including number of layers, hidden units, choice of activation functions, etc, is to determine the network's weights. In contrast to the maximum likelihood techniques which attempt to find a single set of weight values by minimization of a suitable error function, the Bayesian approach considers a probability distribution over weight values. Since the absence of any data, the prior distribution can be assumed to be a zero-mean Gaussian distribution

$$p(w) = \frac{1}{Z_w(\alpha)} \exp(-\alpha E_w) \quad (4.27)$$

where

$$E_w = \frac{1}{2} \|w\|^2 = \frac{1}{2} \sum_{i=1}^w w_i^2 \quad (4.28)$$

$$Z_w(\alpha) = \left(\frac{2\pi}{\alpha}\right)^{\frac{w}{2}} \quad (4.29)$$

4.3.3.2 Posterior Weight Distribution

At first, the prior distribution over weight values is typically given with some general properties such as a broad distribution, or smoothness of the network function. Once the data has been observed, the weights are adjusted to their more probable values through updating the posterior distribution which should be written in the form analogous to equation (4.26)

$$p(w | D) = \frac{p(D | w) p(w)}{p(D)} \quad (4.30)$$

where the denominator is a normalization factor which is

$$p(D) = \int p(D | w) p(w) dw \quad (4.31)$$

For convenience, we assume that the target data is generated from a smooth function with a zero-mean Gaussian noise, the likelihood function $p(D|w)$ is also expressed in the form:

$$\begin{aligned}
 p(D | w) &= \prod_{n=1}^N P(t^n | x^n, w) \\
 &= \frac{I}{Z_D(\beta)} \exp\left(-\frac{\beta}{2} \sum_{n=1}^N \{y(x^n, w) - t^n\}^2\right) \\
 &= \frac{I}{Z_D(\beta)} \exp(-\beta E_D)
 \end{aligned} \tag{4.32}$$

where β is a hyper-parameter which controls the variance of noise, E_D is an error function defined as the equation (4.14)

$$E_D = \frac{I}{2} \sum_{n=1}^N \{y(x^n, w) - t^n\}^2 \tag{4.33}$$

$Z_D(\beta)$ is a normalization factor and is easily evaluated to give

$$Z_D(\beta) = \left(\frac{2\pi}{\beta}\right)^{\frac{N}{2}} \tag{4.34}$$

Based on the chosen a prior distribution and an expression of the likelihood functions, we can obtain the posterior distribution in the form

$$\begin{aligned}
 p(w | D) &= \frac{I}{Z_S} \exp(-\beta E_D - \alpha E_w) \\
 &= \frac{I}{Z_S} \exp(-S(w))
 \end{aligned} \tag{4.35}$$

where $S(W)$ is defined as

$$\begin{aligned}
 S(w) &= \beta E_D + \alpha E_w \\
 &= \frac{\beta}{2} \sum_{n=1}^N \{y(x^n; w) - t^n\}^2 + \frac{\alpha}{2} \sum_{i=1}^W w_i^2
 \end{aligned} \tag{4.36}$$

And

$$Z_S(\alpha, \beta) = \int \exp(-\beta E_D - \alpha E_w) dw \quad (4.37)$$

The most probable weight vector, w_{MP} , can be found by maximising the posterior distribution or equivalently minimising $S(w)$.

4.3.3.3 Posterior Probability of the Hyper-parameters

Although the most probable weight vector, w_{MP} , has been obtained with an assumption that hyper-parameters have been known, we might have little idea of suitable values for α and β . So that in order to search the values of the hyper-parameters, we need to evaluate the posterior distribution of α and β . This is given in the form:

$$p(\alpha, \beta | D) = \frac{p(D | \alpha, \beta) p(\alpha, \beta)}{p(D)} \quad (4.38)$$

In the equation, we can see that the denominator $p(D)$ is independent from the hyper-parameters, so that the maximum posterior distribution for these hyper-parameters α and β is equivalent to maximizing the likelihood term $P(D|\alpha, \beta)$ which is also called the evidence for α and β .

We can express the hyper-parameter evidence in terms of quantities by rewriting the equation to be dependent on α and β explicitly.

$$\begin{aligned} p(D | \alpha, \beta) &= \int p(D | \alpha, \beta) p(w | \alpha, \beta) dw \\ &= \int p(D | \beta) p(w | \alpha) dw \end{aligned} \quad (4.39)$$

By combining the equation (4.36) together with the equations (4.27) and (4.32), we can express the equation (4.39) in the form

$$\begin{aligned} p(D | \alpha, \beta) &= \frac{1}{Z_D(\beta)} \frac{1}{Z_w(\alpha)} \int \exp(-S(w)) \\ &= \frac{Z_S(\alpha, \beta)}{Z_D(\beta) Z_w(\alpha)} \end{aligned} \quad (4.40)$$

We already evaluated $Z_D(\beta)$ and $Z_w(\alpha)$, so that $Z_S(\alpha, \beta)$ should be now evaluated before the evidence $p(D|\alpha, \beta)$ can be obtained. This can be done based on the Gaussian approximation for the posterior distribution of the weights (Mackay 1992d).

$$Z_S(\alpha, \beta) \simeq e^{-S(w_{MP})} (2\pi)^{\frac{W}{2}} |A|^{-\frac{I}{2}} \quad (4.41)$$

where A is the Hessian matrix of the regularized error function evaluated at w_{MP}

$$\begin{aligned} A &= \nabla \nabla S(w_{MP}) \\ &= \beta \nabla \nabla E_D^{MP} + \alpha I \end{aligned} \quad (4.42)$$

Substitute the values of Z_D , Z_w , and Z_S into the equation (4.40), then the log of the evidence is given by:

$$\begin{aligned} \ln p(D|\alpha, \beta) &= -\alpha E_w^{MP} - \beta E_D^{MP} - \frac{I}{2} \ln |A| \\ &\quad + \frac{W}{2} \ln \alpha + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) \end{aligned} \quad (4.43)$$

4.3.3.4 Determination of the Hyper-parameters

Now, we turn to the problem of determining the most probable hyper-parameters by maximizing the posterior, equivalently making derivatives of $\ln p(D|\alpha, \beta)$ with respect to these hyper-parameters be equal to zero. Firstly, we consider the maximum with regard to α :

$$\begin{aligned} \frac{d \ln p(D|\alpha, \beta)}{d\alpha} &= -E_w^{MP} - \frac{I}{2} \frac{d}{d\alpha} \ln |A| + \frac{W}{2} \frac{1}{\alpha} = 0 \\ 2\alpha E_w^{MP} &= W - \alpha \frac{d}{d\alpha} \ln |A| \\ &= W - \sum_{i=1}^W \frac{\alpha}{\lambda_i + \alpha} = \gamma \end{aligned} \quad (4.44)$$

The quantity γ is known as well-determined parameters

$$\gamma = \sum_{i=1}^W \frac{\lambda_i}{\lambda_i + \alpha} \quad (4.45)$$

Similarly, we consider the maximum of the equation (4.43) with respect to β

$$\begin{aligned} \frac{d \ln p(D | \alpha, \beta)}{d\beta} &= -E_D^{MP} - \frac{1}{2} \frac{d}{d\beta} \ln |A| + \frac{N}{2} \frac{1}{\beta} = 0 \\ 2\beta E_D^{MP} &= N - \sum_{i=1}^W \frac{\lambda_i}{\lambda_i + \alpha} = N - \gamma \end{aligned} \quad (4.46)$$

4.3.3.5 Model Comparison

The ultimate goal of the Bayesian methods is to compare different models. In fact, the Bayesian approach is able to automatically penalize highly complex models and so is able to pick up an optimal model without the use of independent data in methods such as cross-validation.

As already indicated, we can compare different models on the basis of their evidence. The evidence of a model can be determined by integrating over the set of network parameters

$$p(D | H_i) = \int p(D | w, H_i) p(w | H_i) dw \quad (4.47)$$

Here, we consider a two layer neural network having M hidden units, then the logarithm of the evidence of the model can be approximately evaluated by the following form (Bishop 1995).

$$\begin{aligned} \ln p(D | H_i) &= -\alpha_{MP} E_W^{MP} - \beta_{MP} E_D^{MP} - \frac{1}{2} \ln |A| \\ &\quad + \frac{W}{2} \ln \alpha_{MP} + \frac{N}{2} \ln \beta_{MP} + \ln M! \\ &\quad + 2 \ln M + 2 \ln \left(\frac{2}{\gamma} \right) + \frac{1}{2} \ln \left(\frac{2}{N - \gamma} \right) \end{aligned} \quad (4.48)$$

As the Bayesian approach to model comparison incorporates a mechanism for penalizing highly complex models, we might expect that the model with the largest evidence would give the best results on unseen data, in other words that it would have the best generalization properties.

4.3.4 Practical Implementation of Bayesian Technique with Levenberg – Marquardt Optimization.

We have covered a lot of ground for the Bayesian technique. Now, we summarize the main steps essential to implement the Bayesian framework within the Levenberg-Marquardt optimization algorithm.

Step 1: The weight values initialized by (Nguyen & Widrow 1990) are very common to utilize in order to decrease the training time. And set $\alpha = 0$ and $\beta = 1$.

Step 2: Use the Levenberg-Marquardt optimization to minimize the total error function $S(w)$ by the equation (4.36).

Step 3: As the error function $S(w)$ reaches the goal (the local minima), then estimate γ , α , β using the equations (4.45), (4.44), and (4.46) respectively.

Step 4: Repeat step 1-3 for several different random initial weight values to find different local minima (in the experiment part, the repeated time is three). Estimate the evidence values for the particular network structure with the different local minima using the equation (4.48).

Step 5: For a collection of different network structures or different hidden unit numbers, repeat steps 1-4, and compare their evidence. The best network chosen from the collection would be the best one on the basis of their evidence.

4.4 Experimental Results

In this section, we will start with some experiments designed to collect data for training the neural network. After that, the training data is used to train the network under the supervision of the Bayesian framework as described above. In fact, the data might be considered as ‘a teacher’ who shows the network how to learn to react to a number of navigating situations. The outcome of the training process is the best trained network for the training data on the basis of the evidence value. In order to demonstrate the efficiency of the found networks in the real - time, we will use these networks to control the wheelchair, and then compare the performances together. We expect that results with the real performances should be consistent with the results drawn from the evidence framework.

4.4.1 Data Acquisition Experiments

4.4.1.1 Data Acquisition Methodology

There are two ultimately important points in sampling data for training network in this application. First, in order to train the network to learn how to react in certain situations, the training data should be obtained from the system created by an experienced user. This is because we are going to create a neural network that learns competent skills to drive the wheelchair the way the experienced user controls the system. Therefore, all samples should be created from that person in order to do so.

Furthermore, acquisition experiments should take place in various environment types and the data should also be sampled in many different scenes in order to guarantee the generalization of the trained network. One of the necessary conditions for good network generalization is that the training cases, or sampling data, must be a sufficiently large and representative subset of the set of all cases that we would like to generalize to. The importance of this condition is related to the fact that there are, loosely speaking, two different types of generalization: interpolation and extrapolation. Interpolation applies to cases that are more or less surrounded by nearby training cases, everything else is

extrapolation. In other words, cases that are outside the range of the training data require extrapolation. Typically, interpolation can be done reliably, but extrapolation is notoriously unreliable. Hence, it is important to have sufficient training data to avoid the need for extrapolation.

Based on these criteria, a number of experiments took place in different locations within Building 1 of the University of Technology, Sydney for convenience. While an experienced operator was using a standard joystick to manually drive SAM to follow various pre-designed paths, software developed in C language in CVI environment was designed to record the surrounding obstacle distances to the wheelchair and control signals at 100ms sampling rate which form a set of input and target, called a pattern. As described in the above section, each pattern includes 37 inputs and 2 outputs. In particular, a set of gathered data can be represented as:

$$D = \{P_1, P_2, \dots, P_N\} \quad (4.49)$$

where D: all data

P_i : a pattern

N: total number of patterns in the data set.

Each pattern comprises of both inputs and outputs illustrated as:

$$P_i = \{I_i, O_i\} = \{\{I_1^i, I_2^i, \dots, I_{37}^i\}, \{v^i, \theta^i\}\} \quad (4.50)$$

Where I_i is an i^{th} input vector of the i^{th} pattern which consists of 37 elements described in detail in the section 4.2.2.2. Similarly, O_i is an i^{th} output vector related to the translational velocity and the rotational speed generated by the user corresponding to inputs at that time. Note that the joystick signals produced by the user are converted to the output vector via the equations (4.3) and (4.4).

4.4.1.2 Collecting training data experiments

As stated in the previous part, all acquisition experiments are carried out in Building 1 of the University of Technology, Sydney because it has various environment types such as a narrow space, walls, doorways, and moving obstacles. Total trials of the acquisition experiment managed for the training purpose are 20 and all of those occurred in different locations. In each trial, the experienced operator manually drove the wheelchair in both directions: from a start point (A) to an end point (B) (way in) - the length of the section AB depends on a particular situation – then, from B to A (way out). While driving the wheelchair, the user was aware that depending on obstacle clearance level and types of environment, the wheelchair should be controlled differently. There were four types of environment selected to carry out the experiments: general obstacle environment, corridor environment, long wall environment, and door environment.

General obstacle environment (5 trials): The level 4 spacious concourse of the building 1 was chosen to implement these experiments where there were tables, chairs, boxes, moving people, etc that were arranged in various ways. Although there were many different obstacles in different directions, the obstacles were relatively far to the wheelchair in comparison to other types. Hence, the navigating strategy was that the operator attempted to remain close to the direction of travel indicated on the machine. At the same time, the wheelchair was intended to drive quickly and smoothly while maintaining a safe distance from surrounding obstacles. The maximum speed was set at 1m/s in these trials.

Corridor environment (5 trials): There were three corridors with different width sizes (3m – 2m – 1.8m) selected for these trials. For corridors, most of the time, the direction of travel shown in the screen pointed towards the centre of the corridor, and the user also smoothly navigated the wheelchair following that direction. The speed of the wheelchair usually remained unchanged during the navigation process. Although it might be up to 1 m/s, specific straight speed largely depended on the size of the corridor objective.

Long wall environment (5trials): A long wall was organized for five trials in which the wheelchair was placed parallel to the wall. Conditions for these trials were identical except the initial distance from the wheelchair to the wall. The initial distances were 0.8 m (for

two trials), 1m (for two trials) and 1.5m (one trial). The goal of the experiments was to drive the wheelchair to run parallel to the wall (3rd to 5th trials) and a small difference between the direction indicated by the machine and the user was acceptable. In the first and second trials, the wheelchair was moved forward but slightly ran away from the wall until the safe distance met as the initial distance (0.8m) was too close to the wall. After that, the wheelchair was navigated in parallel to the wall similar to the 3rd to the 5th trials. The maximum speed of the wheelchair might be up to 1 m/s with the 5th trial while only 0.7m/s with the 1st to 4th ones.

Door environments (5 trials): Typically, door passing task was always the most challenging, and 5 trials relating to five doors with different sizes (1.5m – 1m – 0.9m – 0.8m – 0.7m) were sampled. It is noticeable that the width of the wheelchair plus a safe distance was about 0.64m. In these situations, when moving between two closely spaced obstacles such as the posts of a doorway, the wheelchair slowed down, and centred into the doorway. A slight difference between a steering angle and the desired direction of the machine was usually acceptable.

Samples from the experiments in the different environments were recorded separately to select the best suitable patterns related to the particular environment. The chosen patterns were considered as ones that best match the criteria applied for each environment type. The table 4.1 details a number of patterns collected from the acquisition data experiments. All patterns are utilized for training purposes.

Environment type	Number of Patterns
The Foyer and concourse of the Building 1 with general obstacles such as boxes, chairs, tables, and moving people	300
Corridors with different width sizes	250
A Room with long wall	250
Doors with varied sizes in width	400
Total	1200

Table 4.1: Patterns collected from the acquisition data experiments

4.4.2 Bayesian Neural Network Training Results

For training purposes, a practical Bayesian neural network training toolbox is developed in the Matlab environment based on the knowledge ground discussed in the sections 4.3. After it was developed, the toolbox was used to determine the most probable weight values as well as the evidence value of that feed-forward neural network model. The neural network structure was designed in the training process as follows:

- 37 inputs: 36 inputs obtained from the real-time map and a direction of travel input.
- 2 outputs: translational velocity and rotational angle.
- The hidden node number of the feed forward network was set to vary from 1 to 10. Consequently, we are able to assess the performance of each model based on the collected data.

Because the result of the training process is quite sensitive with the network's weight initializations, we are going to mitigate the effect by using multi-training times for each

model network. In particular, the training of each structure is repeated 3 times and the evidence for that structure is evaluated by the average values of the three times.

The figure 4.7 shows the result of the assessment of the models. It can be seen that the average log evidence of the models increases as the number of hidden nodes increases until the structure has four hidden nodes. Then, it starts to decline as the hidden node number continues to increase. Based on individual training, the network with 4 hidden nodes obtained the highest evidence value although the differences in the highest evidence values achieved by the neural network solutions using 3, 4, 5 and 6 hidden nodes are quite small. To clearly confirm the network having the highest evidence value would have the best generalization, the trained networks should be used to navigate the wheelchair in real –time in order to compare the performance.

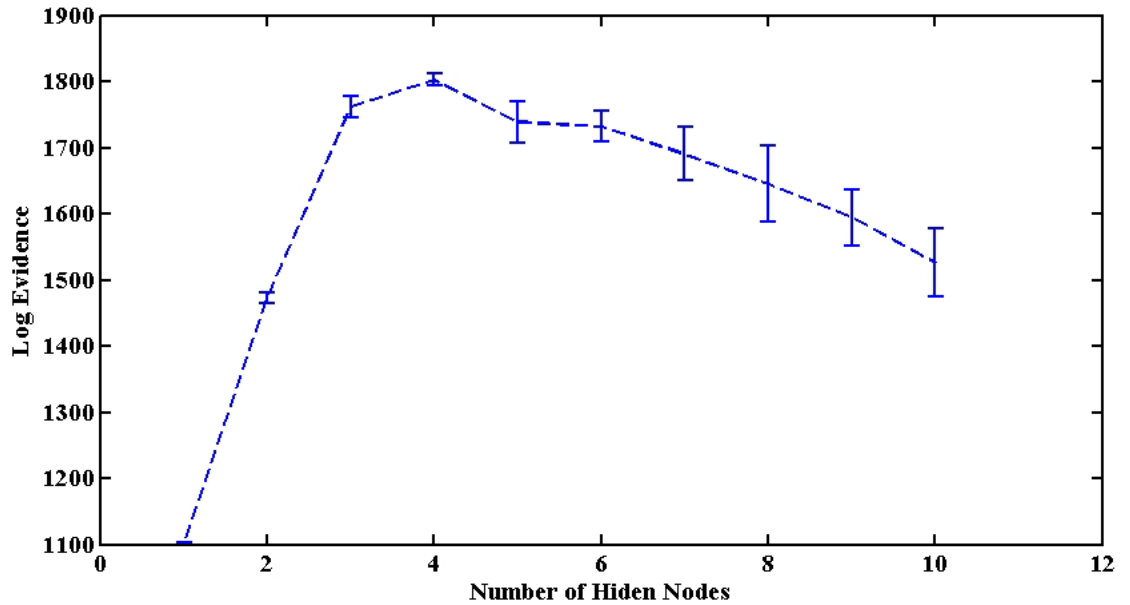


Figure 4.7: Estimation of log evidence for neural networks with different hidden nodes

In order to compare the generalization of different networks in navigating the wheelchair system to avoid obstacles, these structures and their weight values were extracted into a text file, and then loaded into the software to control the wheelchair. The following two real navigation experiments were designed for comparison purposes. The first experiment was about examining these neural network structures to control the wheelchair in unknown and

clustered environments. The other experiment was to evaluate the performance of the networks to passing narrow doors.

4.4.3 Experiment 1: Navigating SAM in Unknown and Dynamic Environments

4.4.3.1 Experiment Description

The experiment was conducted in the laboratory room in which both static obstacles (walls, chairs, a couch) and dynamic obstacles (person walking) were present as shown in the figure 4.8. It is noticeable that sampling data for the neural network training was not taken in this room. In other words, this environment was considered as the unknown to all the trained neural network structures that were used to run the experiment.

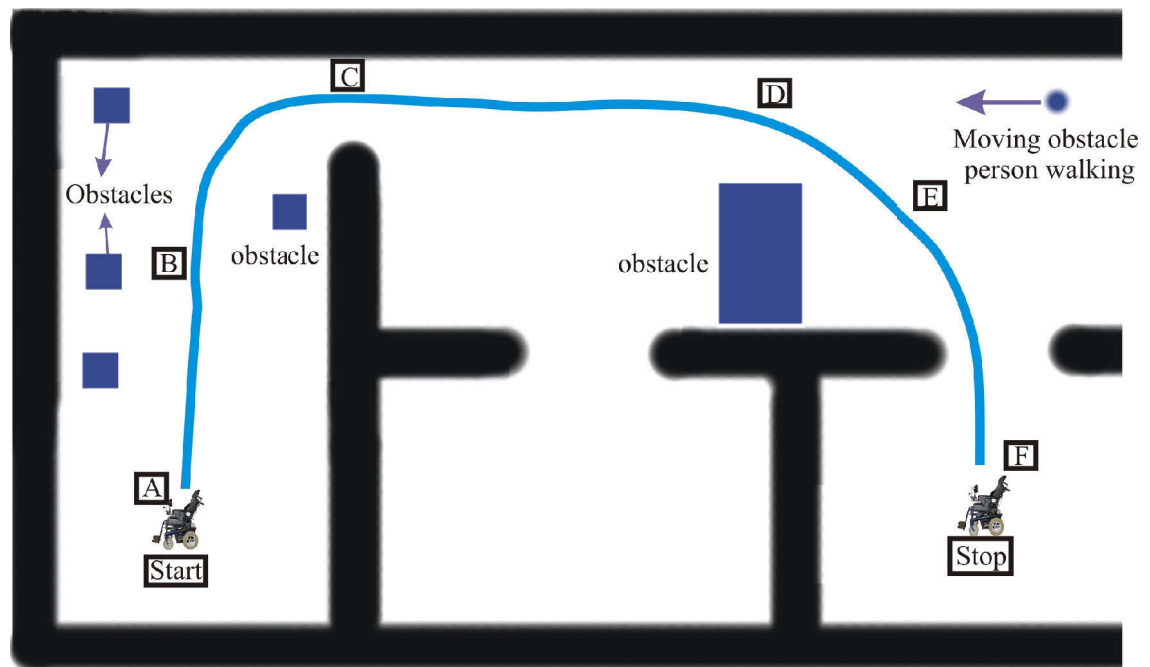


Figure 4.8: Each neural network structure is asked to autonomously navigate the wheelchair following the blue line

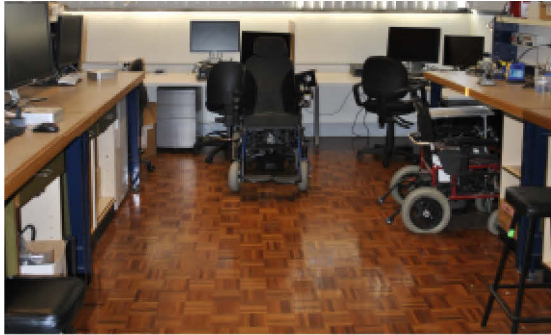
The goal of this experiment was in order to evaluate the performance of the different neural network structures having 3, 4, 5 and 6 hidden nodes with the highest evidence value in autonomously controlling the wheelchair to complete the trajectory A - B - C - D - E - F as the blue line. The test course was deliberately arranged in various sections with different types of environment. As shown in the figure 4.8, the sections AB and DE has general obstacles like chairs, tables or a moving person, the section CD contains a wall, and the SAM wheelchair was expected to follow that wall. Finally, the sections BC and EF have narrow spaces – doors with the width size about 1.2m.

The structure and weight values of testing network models, one after the other, were loaded into the system to take charge of controlling the wheelchair. Starting at the begin point A, each testing network model automatically drove the SAM wheelchair to move to the end point F without any human intervention in 10 trials. For the evaluation purpose, the velocity of the wheelchair as well as the number of collisions would be recorded for comparisons.

4.4.3.2 The Trajectory Observation

Overall, the SAM wheelchair under the control of the testing neural network models was able to complete the designed trajectory. Although some collisions occurred with the control of the trained networks with 3, 5 and 6 hidden nodes, the trained networks had the capability to real-time navigate the SAM wheelchair in the unknown and dynamic environment. As permitted to be moved at the point A, the vehicle started to detect the surrounding free spaces. In this case, only a front free space was found, the networks generated the control signals to move the SAM wheelchair forward, and then turn the wheelchair into the door C. At point C, the system decided to move forward as that free space had greater probability than the door on the right side. By institution, the SAM wheelchair under the control of the networks moved parallel with the left side wall. At point D, SAM detected a moving person, so that the network slightly steered SAM on the right side to avoid the person. Finally, the system was driven into the door in the front and stopped at the point F

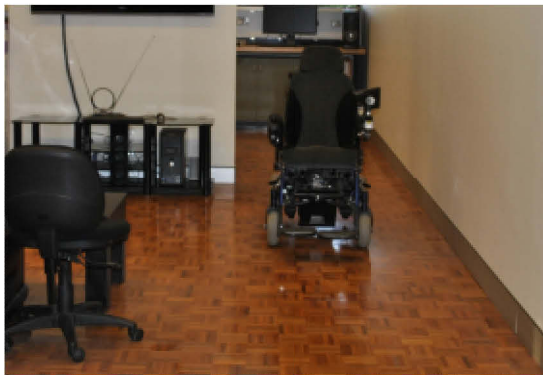
The figure 4.9 shows an example of the test run that was performed by the trained neural network with 4 hidden nodes.



The wheelchair starts at point A



The wheelchair is approaching the door (section BC)



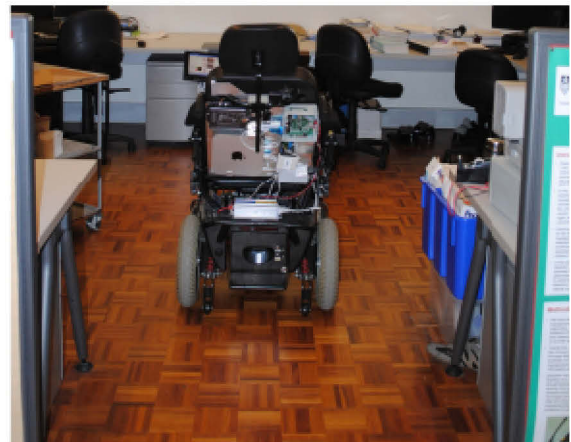
The wheelchair just passes the door



The wheelchair is at the point D



The wheelchair is reaching to the other door (section EF)



The wheelchair is at the point F

Figure 4.9: An example of a test run conducted by a neural network with 4 hidden nodes

4.4.3.3 The Comparison between the Neural Network Model Performances

While performing the navigating task, software recorded some navigating parameters for comparison purposes. The parameters included average speed of the wheelchair in each section AB, BC, CD, DE and EF and a number of collisions during navigation of each model. A collision was considered when the wheelchair physically touched any obstacles while navigating. If there is a collision which leads to stop the system, then the test run is marked as failure and all parameters are eliminated. The result of the measurement is shown in the table 4.2

Neural network with	Average speed per AB (m/s)	Average speed per BC (m/s)	Average speed per CD (m/s)	Average speed per DE (m/s)	Number of collisions
3 hidden nodes	0.71±0.12	0.34±0.11	0.69±0.09	0.42±0.13	2
4 hidden nodes	0.8±0.039	0.35±0.031	0.67±0.018	0.36±0.038	0
5 hidden nodes	0.75±0.15	0.38±0.18	0.68±0.09	0.38±0.16	4
6 hidden nodes	0.82±0.13	0.4±0.10	0.69±0.12	0.45±0.12	3

Table 4.2: .The results of the performance of the neural network with 3,4,5 and 6 hidden nodes when controlling wheelchair in ten trials

In terms of the velocity under the control of the testing neural networks, the table 4.2 presented the mean $\pm$ standard deviation of the average speed in the different sections of each network model in the 10 tested trials. It can be clearly seen from the standard deviation of the average speeds that the neural network with 4 hidden nodes produces more stable speed across the sections than that with 3, 5, and 6 hidden nodes. This means that the behaviour of SAM produced by the 4 hidden node network would be better than those of other networks.

In terms of collisions, the neural network with 5 hidden nodes produced the poorest performance with 4 collision times out of the 10 trials, and the network with 4 hidden nodes navigated SAM without any collision in all trials. Despite the fact that collisions occurred with the control of the trained network with 3,5 and 6, most of those collisions just slightly touched the door frame and none of those caused the system to stop.

In order to see the adaptation of the network with 4 hidden nodes under the different environment types, the pair-wise comparisons of the average speeds across the sections were performed using the standard t-test. The t-test showed that the average speed of the section AB was significantly faster than that of the section BC ($p = 0.0017$), section CD ($p = 0.0017$) and DF ($p = 0.0017$). Also the speed of the section CD was significantly faster than that of the section BC ($p = 0.0017$) and DF ($p = 0.0017$). However, there is no difference between the average speed of the section BC and DF. These comparisons prove that the neural network with 4 hidden nodes was able to recognise the differences in the environment arrangement, and produced the different control signals accordingly.

4.4.4 Experiment 2: The door passing experiment

4.4.4.1 Experiment Description

Another experiment was conducted to demonstrate the ability of the obstacle avoidance neural networks in driving the wheelchair to pass between closely spaced obstacles such as a door. In this experiment, a door having a changeable width from 0.65m to 1.2m (the wheelchair's width is 0.64m) was used as depicted in the figure 4.10.

With each width size of the door, the SAM wheelchair with the selected networks began at about 1.5 - 2m away from the door opening, and started to pass through the door. The network with specific hidden nodes repeatedly guided the wheelchair through the door size 10 trials for consistency. For a performance comparison purpose, the wheelchair was also steered by an experienced human using a joystick and the performance of the user was assumed as a "perfect performance". The percentage of successful passing per each door's width, which was defined as the ratio between the successful trials and the total trials multiplying with 100, was recorded in the figure 4.11.



The wheelchair is about the start point



The wheelchair is approaching the test door

Figure 4.10: The door passing task is to assess the performance of the neural network with different structures

4.4.4.2 Door Passing Results

The figure 4.11 showed that all networks were able to drive the wheelchair to pass through the door with 1.2m width or more with a 100% success rate. Note that these results were only applied for the wheelchair's initial position directed to the centre of the door; if the wheelchair is placed with an acute angle to the door, the result would be certainly lower.

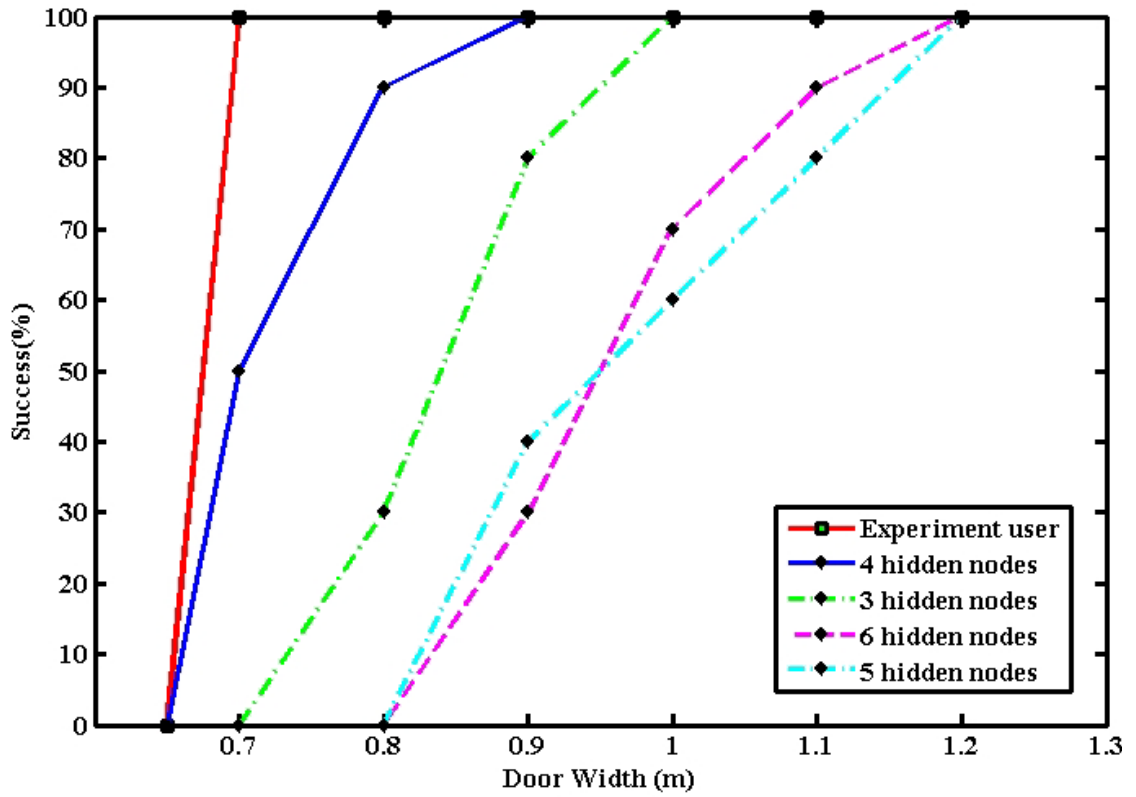


Figure 4.11: Results of the performance of the neural network models for the door passing task

From the plot, we see that the network with 4 and 3 hidden nodes were clearly better than that with 5 and 6 hidden nodes, in which, the neural network with 4 hidden nodes achieved the best performance for the passing door task. The percentage of successful passing was recorded: the doorways of width 70 cm with 50%, 80 cm with 90%, 90cm or more with 100%. This result was quite consistent with the above navigation experiment in which this network generates more stable control signals when the wheelchair is entering doors than other structures.

Comparing with the highest performance developed in our previous report (V.Nguyen. et al. 2012) which found the 6-hidden node network to be the most optimal, it can be obviously seen that the door passing performance of SAM with 4 - hidden node neural network was significantly improved. With this advanced method, the network only needs four hidden nodes to navigate the wheelchair to avoid obstacles. That means the computing

time during navigating is shorter, resulting in the system responding faster. This might explain why the results of using both networks were quite similar for real-time navigation in the laboratory room where obstacles were relatively far or doors are quite wide (approximately 1.2m). However, the four hidden node network improved door passing performance of the wheelchair when the SAM wheelchair was able to pass the doorways of width 80 cm with 90% and 90cm with 100% , while with the six hidden node network, SAM was only able to cross those doorways with 50% and 80% respectively.

4.5 Discussion

This chapter describes the development of an autonomous navigation using Bayesian neural network. The proposed method has been demonstrated to work very well in unknown environments and to be able to drive the wheelchair passing through very narrow spaces such as the door passing task.

A crucial contribution of this chapter is to implement a Bayesian neural network training tool box which assists to increase the generalization of network during the training process. Although initially introduced in the 1990s, little has been done to implement this method for practical applications. The training and results presented in this chapter imply that our technique, through estimating Hessian matrix and the Levenberg-Marquardt optimization under the supervision of Bayesian theory, can be effectively used to find both the most optimal structure and the most probable weight values of a neural network given a training data set. The network with the higher estimated evidence provides the better real - time navigation performance.

Another important contribution in this chapter is to provide an effective obstacle avoidance method that relies on the machine paradigm. Although there have been some reports in the literature review, none of them is able to provide a detailed procedure for the practical implementation or real-time experimental results. The result indicates that the trained network based on the designed patterns is able to adapt to the changes of the environment. In particular, it is able to recognize the environment type and provide the control that is

appropriate to it. This provides a great improvement for the assistive system which requires the system to not only effectively avoid obstacles, but also to satisfy the psychology acceptability, safety and robustness as mentioned in the chapter 2.

Until this point, the SAM wheelchair is developed to automatically detect free spaces and navigate without collisions in the unknown and dynamic environment. The autonomous abilities are prerequisite to provide the navigation assistance to the user with impairments. However, a completely autonomous system is not the aim of this project, because most users would avoid a system in which the user is always being led by the machine. Obviously, it is necessary to develop a shared control strategy in order to integrate the users into the system, and to allow them involvement in the navigation process.

Chapter 5

Shared Control Strategy Based on Bayesian Theory for a Semi-Autonomous Wheelchair

5.1 Introduction

The main difference between a mobile robot and an intelligent wheelchair is that the intelligent wheelchair has two control subjects: an assistive autonomous navigation controller and the user with disabilities seeking mobile support. Thus, it is necessary to obtain the cooperation between the human user and the wheelchair controller in the achievement of the mobility task (Mazo 2001; Parikh et al. 2007; Simpson 2005; Urdiales et al. 2011). From a technical perspective, the collaboration between the person and the machine might enhance the robustness, safety, and the manoeuvrability for the system. For instance, the user with weak physical but generally great cognitive possibilities prevents the system from colliding with undetected obstacles due to the limitations of the hardware devices. Apart from the technical interest, it is psychologically imperative to involve the user in the system control (Nisbet 2002b). The user would be reluctant to use the system which is always led by the machine rather than the human commands. In short, in such an assistive application as this, the intelligent wheelchair with complete autonomy does not seem to be an ideal end to reach.

Throughout chapter 3 and chapter 4, the SAM intelligent wheelchair has been equipped with robust algorithms to find free-spaces in the unknown and dynamic environment and automatically navigates into the free spaces by using the trained Bayesian neural network. The SAM wheelchair with the autonomous neural network's driving behaviour is usually

driven in the optimal trajectories, and protected from colliding. In this chapter, the SAM system will be developed further by integrating the user inputs with the autonomous behaviours. The goal of the proposed shared control strategy is to allow the user to be actively involved in the navigation process as much as possible while still providing the assistance needed to achieve the manoeuvre safely.

Unlike many other shared control methods which use internal mapping of the surroundings to predict the user's intent and implement the shared control, the proposed method for the SAM wheelchair is investigated based on the real-time map of the local environment and the observed user control data. This approach is certainly much more advantageous than the other methods in the intelligent wheelchair application because it allows the system to operate in unstructured and dynamic environments rather than depending on the internal maps. Although having many advantages, using only the onboard information to estimate the user's intent has its challenges when dealing with the uncertainty rising from both the operator and the environment.

The uncertainty of the user's inputs rises from the fact that a user with impairments is unable to properly provide control over the vehicle. Meanwhile, the uncertainty in the environment information comes from the unknown environment factors and the limitations of the measure sensors. To solve the problem associated with the uncertain information, we will propose to use probability to represent the uncertainty, then apply the Bayesian interpretation of probability to quantify our expression of the uncertainty and make precise revisions in the light of newly observed data. Subsequently, we will be able to take optimal shared control actions or decisions as a consequence.

This chapter will be organised as follows. After the introduction, the shared control strategy architecture and its practical implementation will be explained in detail. Finally, two experiments will be presented in this chapter. The first experiment is to demonstrate that the system using the Bayesian shared control framework is able to make the most appropriate shared control action to navigate in real-time. The second experiment implemented with a group of able-bodied participants is to evaluate the performance of the semi - autonomous system on the basis of quantitative and qualitative measurements.

5.2 Advanced Shared Control Architecture for Semi-autonomous Intelligent Wheelchair

In the application of mobile robotics to assist people with disabilities, the control structure should be proposed in order to optimize the collaboration between a person who is weak physically and the machine with complementary capacities. On one hand, the user who has reduced physical capacities seeks the mobile assistance that has to feel natural, while on the other hand, the machine has capabilities to implement precise control signals to manoeuvre the vehicle in unknown and dynamic environments without collisions.

The figure 5.1 presents the architecture of the shared control strategy between the user and the machine for the SAM. As shown in the figure, the SAM system is constituted from three main parts:

User – wheelchair interfaces: This is the key component that assists in the exchange of the information between the user and the machine. In order to accommodate the target group of users with a great diversity of injury levels, four different interfaces, including both joystick - based and hands-free technologies have been created to equip the SAM wheelchair. These interface devices can be easily added or removed from the system as desired. For inputting commands, the user has four alternative interfaces: a standard joystick, iphone/ipad, head movement interface (HMI), and an advanced brain computer interface (BCI). In accordance with the needs as well as the level of injury of the particular user, he can choose one of these inputting methods. Through the interaction between the user and the interface, the SAM system would be able to predict and follow the user's intention.

Environment interpretation: As having been stated in chapter three, the SAM's controller is able to detect the surrounding obstacles, thus determining a number of free spaces to navigate. The free space determination is on the basis of the measurement from the sensor system as well as the capability of passing doors. The information about the local free space would play an important factor in making suitable decisions of the shared control with the user.

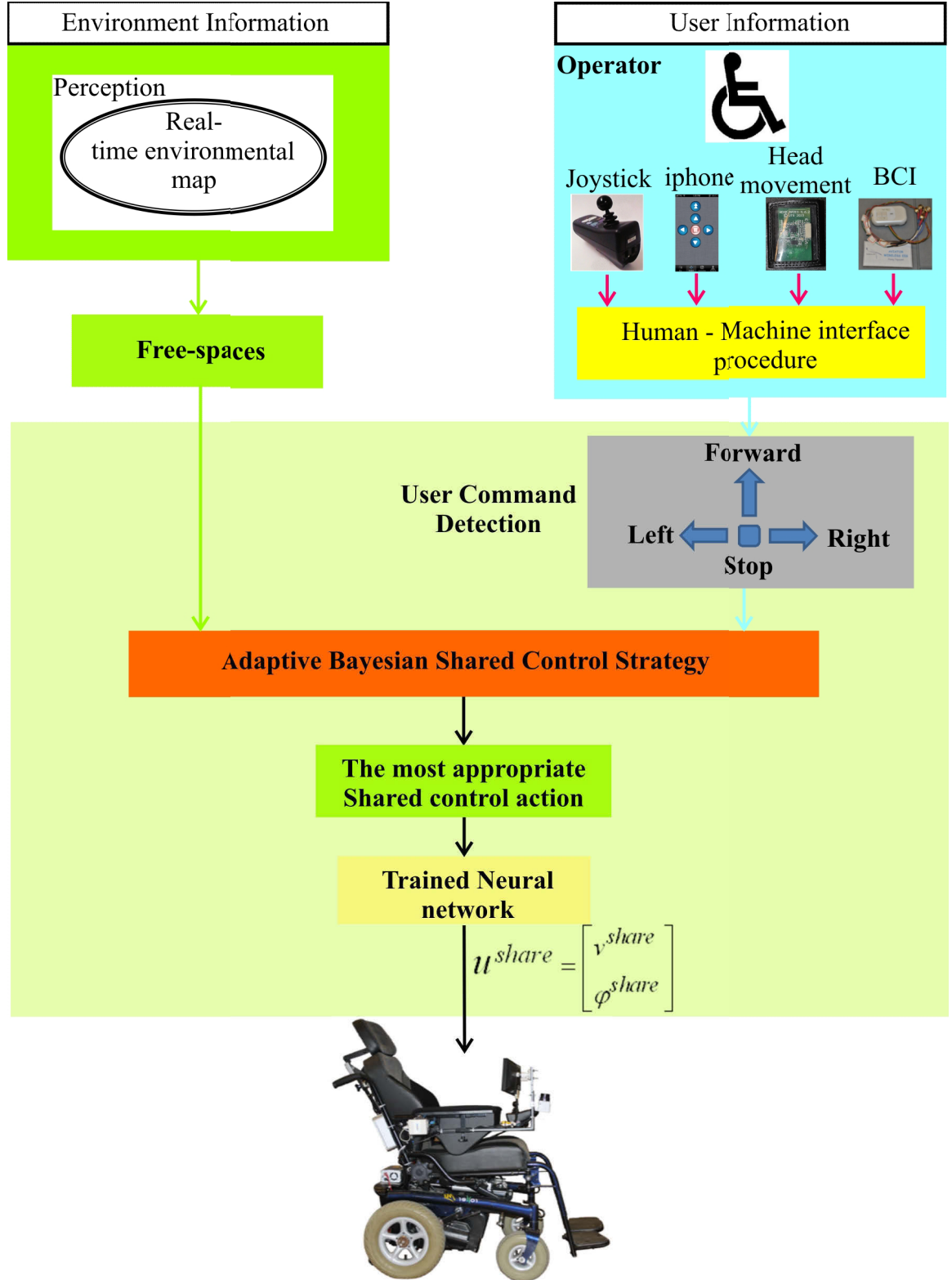


Figure 5.1: The Shared control architecture of the SAM intelligent wheelchair

Adaptive Shared control strategy: This part is to share the control between the user and the assistive navigational controller. It is motivated by the fact that the system should assist the user by relieving the tediousness of control, overcoming the safety control issues associated with driving and importantly, the system should follow the user's intentions. The goal of the shared control framework is that the user is permitted to be actively involved in the navigation process while still providing the navigation assistance needed to achieve the optimal and safe manoeuvre. It is obvious that the goal is only achieved if the proposed method is able to select the optimal action which is appropriate to both the user's intent and the environment information.

As shown in the figure 5.1, the shared control decision is made from combining the user's commands and the available free spaces at that moment. In particular, the user expresses his intentions by means of electrical signals via the user – machine interface. As these signals contain noise, they must be processed to predict the user control command such as go forward, turn left, turn right or stop. Based on the command prediction, adaptive Bayesian shared control framework will be applied in order to combine the uncertain information from both the user and environment to find the most likely action. The most appropriate action can be understood as selecting the free space for navigation which is consistent with the obtained user command and the environment information. After the shared control action is made, the trained Bayesian neural network developed in chapter 4 is used to generate the control signal which is

$$\mathbf{u}^{share} = \begin{bmatrix} v^{share} \\ \phi^{share} \end{bmatrix} \quad (5.1)$$

where v^{share} : is the translational velocity generated by the neural network.

ϕ^{share} : is the rotational velocity generated by the neural network.

Obviously, the command $\mathbf{u}^{share}$ is the result of the collaboration between the user and the machine. The user's contribution is in deciding the free space to navigate, while the assistive navigation controller provides its precise computation in manoeuvring the wheelchair.

5.3 The implementation of Advanced Bayesian Shared Control Strategy

The efforts to implement an effective share control for the SAM system have been reported in our papers (Nguyen et al. 2011; Nguyen et al. 2013). The main goal of the advanced shared control strategy is to achieve the best predictive performance on the uncertain observed data from the user and the environment. It is obvious that the use of probability to represent the uncertainty, however, is not an ad-hoc choice, but is inevitable if we are to respect common sense while making rational coherent inferences.

5.3.1 Adaptive Shared Control Strategy Based on Bayesian Inference

In this section, an adaptive shared control based on the Bayesian theory is developed to integrate the user's inputs with the autonomous behaviours in the SAM wheelchair system. The proposed method adapts with the changes of information obtained from the user as well as the environment so as to share the control between user and machine appropriately. In order to do so, both the information sources which are always uncertain must be investigated carefully to find the user's intention and make the shared control decisions. The uncertainty of the user's inputs rises from the fact that the users with impairments are not able to use the human – machine interface properly. As a result, the produced control signals fail to represent their intents, and cause the oscillation or even collisions during the navigation. Meanwhile, uncertainty in the environment information comes from the limitations of the measure sensors and from the unexpected events because the SAM wheelchair operates in the unknown and dynamic environments.

To deal with the uncertain information, it would be natural to use probability to represent uncertainty, and then the Bayesian interpretation of probability to quantify our expression of the uncertainty and make the precise revisions of the uncertainty in the light of new evidence, as well as subsequently to be able to take optimal actions or decisions as a consequence. Bayes' theorem, which takes the form

$$P(X / O) = \frac{P(O | X)P(X)}{P(O)} \quad (5.2)$$

where

X: is a hypothesis about the action of the SAM intelligent wheelchair

O: is the observed data

P(X|O): is the posterior probability that determine the probability of the hypothesis X being true given the new data O is observed.

P(O|X): is the likelihood distribution that express how probable the observed data set is for different settings of the hypothesis X.

P(X): is the prior probability that is the probability of the hypothesis X being true before observing the new data O

P(O): is the normalization constant which ensures that the posterior distribution on the left-hand side is a valid probability density and integrates to one.

The equation (5.2) allows us to update the posterior probability of a number of hypothesized actions of the SAM wheelchair based on the new observed data, and then compare the posteriors together to find the most appropriate action for the wheelchair which has the highest posterior probability. If doing so, it is certain that the selected action would not only be the best suitable with the user's intent, but would also be in harmony with the environment information.

Let us represent the equation (5.2) as the full evolution of the time. Assume that up to the time t, the wheelchair controller has received a stream of the observed information

$$\begin{aligned} O_{1:t} &= \{o_1; o_2; \dots; o_t\} \\ &= \{(u_1, w_1); (u_2, w_2); \dots; (u_t, w_t)\} \end{aligned} \quad (5.3)$$

where: $o_t = (u_t, w_t)$ is the observed data of the computational time t which consists of the user control data u_t and the free space information w_t

Corresponding to the stream of the observed information, SAM has a set of selected actions up to time t

$$X_{1:t} = \{x_1; x_2; \dots; x_t\} \quad (5.4)$$

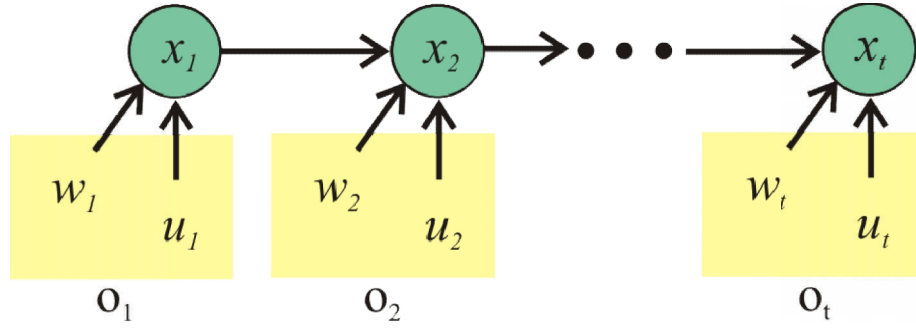


Figure 5.2: The Bayesian network that SAM uses to estimate the most probable shared control actions

The evolution of the actions $X_{1:t}$ and the observed data $O_{1:t}$ is depicted as the figure 5.2. By combining the equations (5.2), (5.3), and (5.4), the actions of the SAM wheelchair with respect to time t are calculated as follows

$$\begin{aligned} P(X_{1:t} | O_{1:t}) &= \frac{P(o_t | X_{1:t}, O_{1:t-1}) \times P(X_{1:t} | O_{1:t-1})}{P(o_t | O_{1:t-1})} \\ &= \frac{P(o_t | X_{1:t}, O_{1:t-1}) \times P(x_t | X_{1:t-1}, O_{1:t-1}) \times P(X_{1:t-1} | O_{1:t-1})}{P(o_t | O_{1:t-1})} \end{aligned} \quad (5.5)$$

Apply the Marko assumption about the complete state that postulates that the past and future data are independent if one knows the current states, so that the equation (5.5) can be reduced by

$$P(X_{1:t} | O_{1:t}) = \frac{P(o_t | x_t) \times P(x_t | x_{t-1}) \times P(X_{1:t-1} | O_{1:t-1})}{P(o_t | O_{1:t-1})} \quad (5.6)$$

The first term on the right-hand side $P(o_t | x_t)$ is called the measurement probability which represents the probabilistic law according to which measurements o_t are generated given the action x_t . The second right-hand side $P(x_t | x_{t-1})$ is known as the transition probability which specifies how the hypothesized actions evolve over time.

The posterior probability distribution of the action for the SAM wheelchair at each computation time step t can be estimated by multiplying the measure probability of the time t and the integration of the prior probability across the action domain.

$$P(x_t | O_{1:t}) = \frac{P(o_t | x_t)}{P(o_t | O_{1:t-1})} \int_{x_{t-1}} P(x_t | x_{t-1}) \times P(x_{t-1} | O_{1:t-1}) \times dx_{t-1} \quad (5.7)$$

If the number of hypothesized actions of SAM is limited, then the equation (5.7) in the discrete variables would become

$$P(x_t | O_{1:t}) = \frac{P(o_t | x_t)}{P(o_t | O_{1:t-1})} \sum_{x_{t-1}} P(x_t | x_{t-1}) \times P(x_{t-1} | O_{1:t-1}) \quad (5.8)$$

From the equation (5.8), to determine the probability of a certain action x_t , its probability must be determined first given the previous actions x_{t-1} . That probability is calculated by multiplying the transition probability $P(x_t | x_{t-1})$ and the prior probability $P(x_{t-1} | O_{1:t-1})$, then summing over the whole domain of possible hypothesized actions

$$P(x_t | O_{1:t-1}) = \sum_{x_{t-1}} P(x_t | x_{t-1}) \times P(x_{t-1} | O_{1:t-1}) \quad (5.9)$$

This step is usually named as the prediction step. Then, the next step, called the measurement update, involves updating the observation model $P(o_t | x_t)$

$$P(x_t | O_{1:t}) = \frac{P(o_t | x_t)}{P(o_t | O_{1:t-1})} \times P(x_t | O_{1:t-1}) \quad (5.10)$$

The denominator in the equation is independent of the actions that we are tracking, thus, its function is only to ensure that the resulting posterior is the proper probability density function.

The formulas (5.9) and (5.10) theoretically describe the probabilistic framework to estimate the posterior probabilities of the SAM wheelchair's actions as it obtains new observed data comprising the user's command and the environment measurement. To make this framework feasible to implement in practice, the domain of actions for the SAM wheelchair

must be specified, and this typically depends on the need as well as the driving capabilities of the targeted user group of the vehicle.

We aim to provide the mobile support to the users with disabilities who generally lack proper driving skills. As a result, the fuzzy and inconsistent control data, obtained via the human - machine interface from the user group, is only used to predict some general directions to navigate such as go forward (F), turn left (L), turn right (R), or stop (S). Corresponding to the commands, the SAM actions x_t at the computational time step t might be F, L, R, or S. With the defined action domain, the two – step update can be rewritten as follows:

$$\begin{aligned} \text{Prediction: } P(x_t^k | O_{1:t-1}) &= \sum_{j=F,R,L,S} P(x_t^k | x_{t-1}^j) \times P(x_{t-1}^j | O_{1:t-1}) \\ \text{Measurement update: } P(x_t^k | O_{1:t}) &= \frac{P(o_t | x_t^k)}{P(o_t | O_{1:t-1})} \times P(x_t^k | O_{1:t-1}) \end{aligned} \quad (5.11)$$

The superscript k and j are used to indicate that the SAM wheelchair takes the action k and j at the time t and $t-1$. They must be one of the four pre-defined actions: F, R, L, or S.

5.3.2 Likelihood Distribution Modelling

Both the transition probability $P(x_t|x_{t-1})$ and the observation probability $P(o_t|x_t)$, also known as the likelihood distributions, have to be identified before the posterior probability $P(x_t|o_t)$ can be calculated. In this section, the likelihood distributions will be modelled on the basis of the observed data.

5.3.2.1 The Transition Probability

The transition probability $P(x_t|x_{t-1})$ should express how likely a certain change in the action x_{t-1} to the action x_t is between two consecutive computation time steps. As the prediction step is processed before the new data is observed, the transition probability $P(x_t|x_{t-1})$ must

be determined based on the knowledge of the previous measurements. The fact is that the SAM wheelchair navigates in the unstructured and dynamic environment, as a result, the frequent changes in the navigation intent are very likely to occur, not only because of sudden changes in the driver's intent, but also because of the changes in the surrounding environment such as people walking by.

Unless there is a priori evidence that a certain navigation action is more probable, there is no reason to believe that the action would deserve having the higher probability. The priori evidence can be seen as the environmental cues and the state of the system at the previous time step. Possibly, the transition probability $P(x_t|x_{t-1})$ might be modelled relying on the prior environment information w_{t-1} and the SAM's belief of the current action. For instance, if the SAM wheelchair detected that there was only forward free space at (t-1), then the vehicle should believe that it is likely to go forward or stop in the next step, and it is very unlikely to be able to turn left or turn right because these actions are not supported by the environment measurement. The belief can be expressed as follows

$$P(x_t^k | x_{t-1}^j) = \begin{cases} 0.6\gamma & \text{for } k = j \text{ and } P(w_{t-1}^k) \geq 0.5 \\ 0.3\gamma & \text{for } k \neq j \text{ and } P(w_{t-1}^k) \geq 0.5 \\ 0.1\gamma & \text{for } P(w_{t-1}^k) < 0.5 \end{cases} \quad (5.12)$$

The equation (5.12) shows that the system favours its action unchanged between the two consecutive time steps. The w_{t-1}^k is the environment measurement at time step (t-1) that supports for the predicted action k at time step t. For instance, the predicted action is F, then, $P(w_{t-1}^k)$ is the probability of the forward free space. Finally, the predicted action must be bounded by the predefined action domain, so that the coefficient γ is used to make sure the transition probability density function is normalized. In other words,

$$\sum_{k=\{F,R,L,S\}} P(x_t^k | x_{t-1}^j) = 1 \quad (5.13)$$

5.3.2.2 The Measurement Probability

The measurement probability $P(o_t|x_t)$ suggests that the SAM wheelchair is relatively reliable in detecting free spaces w_t in the surroundings and in predicting the user command u_t given that the action x_t is selected. Thus, it is reasonable to identify the measurement probability $P(o_t|x_t)$ as a mixture distribution of the environment measurement probability and the user command probability by

$$P(o_t | x_t^k) = \eta \times P(w_t^k) \times P(u_t^k) \quad (5.14)$$

Similarly, η is used to ensure that the measurement probability $P(o_t|x_t)$ is a proper probability density distribution.

$$\sum_{j=(F,R,L,S)} P(o_t | x_t^j) = \sum_{j=(F,L,R,S)} \eta \times P(w_t^j) \times P(u_t^j) = 1 \quad (5.15)$$

Combine the equations (5.14) and (5.15), the measurement probability is determined as follows

$$P(o_t | x_t^k) = \frac{P(w_t^k) \times P(u_t^k)}{\sum_{j=(F,L,R,S)} P(w_t^j) \times P(u_t^j)} \quad (5.16)$$

The environment measurement probability $P(w_t^k)$ can be determined as the probability of free space that supports to the action k at the time step t . It is noticed that the S action happens without the support of any free spaces; therefore, the measurement probability for this action is usually set to be a constant. The probability of the user command $P(u_t^k)$ is the probability of the command prediction. In the next section, we will propose the Bayesian neural network to recognize the user's control signals in order to predict the user commands. The accuracy of the command detection would determine the probability of the user command $P(u_t^k)$.

5.3.3 Bayesian Neural Network for Head Movement Detection

The SAM wheelchair is designed to work with a number of the human – machine interfaces as introduced in the section 3.3.3 in chapter 3. Each interface type, the method to process the user's signals to predict commands, are different. As a result, the accuracy as well as the user command prediction probability $P(u_t|x_t)$ are also varied across the interfaces. To make the method determining the probability $P(u_t|x_t)$ more concrete, we will focus on processing the control signals of the head movement interface to predict the user command in this section.

The sensor detects the head movement in the form of tilt on two axes, forward/back and left/right. In other words, the received data typically include two channels FB and LR representing the tilt angle differences between the reference accelerometer that was attached behind the chair and the head accelerometer. The FB indicates the forward and backward tilt and the LR suggests the left and right tilt. A tilt of about 10 to 40 degrees produces a range of zero and to maximum. Both values FB and LR are not linear with the angles α , β between the wheelchair and the head accelerometer, but they are related together by the following formulas:

$$\begin{aligned} FB &= A \sin \alpha \\ LR &= B \sin \beta \end{aligned} \quad (5.17)$$

where A, B are magnitudes, and α , β are zero corresponding to the vertical. These magnitudes can be estimated by experimental tests which let α , β try some different values, and record FB, LR values to interpolate A and B. From tests:

$$\left. \begin{aligned} &\alpha, \beta = \pm 5 (\text{deg}) \quad FB, LR = \pm 557 \\ &\Rightarrow A = B = \frac{FB}{\sin \alpha} = 6390.85 \\ &\alpha, \beta = \pm 40 (\text{deg}) \quad FB, LR = \pm 4454 \\ &\Rightarrow A = B = \frac{FB}{\sin \alpha} = 6929.2 \end{aligned} \right\} \Rightarrow \begin{cases} FB = 6660 \times \sin \alpha \\ LR = 6660 \times \sin \beta \end{cases} \quad (5.18)$$

The following threshold - based calculation shows how to convert FB, LR into the appropriate voltages to manually control the wheelchair.

Forward – Reverse:

$$FB = 6660 \sin \alpha; \quad FB_0;$$

$$\text{for } FB \geq FB_0, \quad a = FB - FB_0$$

$$\text{for } -FB_0 < FB < FB_0, \quad a = 0$$

$$\text{for } FB \leq -FB_0, \quad a = FB + FB_0$$

$$X = \left(\frac{a}{8192} \times 2.5V \right) + 2.5V \quad (5.19)$$

Left – Right

$$LR = 6660 \sin \beta; \quad LR_0;$$

$$\text{for } LR \geq LR_0, \quad a = LR - LR_0$$

$$\text{for } -LR_0 < LR < LR_0, \quad a = 0$$

$$\text{for } LR \leq -LR_0, \quad a = LR + LR_0$$

$$Y = \left(\frac{a}{8192} \times 2.5V \right) + 2.5V \quad (5.20)$$

In practice, the threshold values FB_0 and LR_0 are very different as the ‘at rest’ position of the system significantly changes. Therefore, an adaptive calibration system is used to ensure that the sensor is re-calibrated. After being re-scaled by the equation (5.19) and (5.20), the outputs of the head movement sensor are demonstrated in the figure 5.3 and the figure 5.4

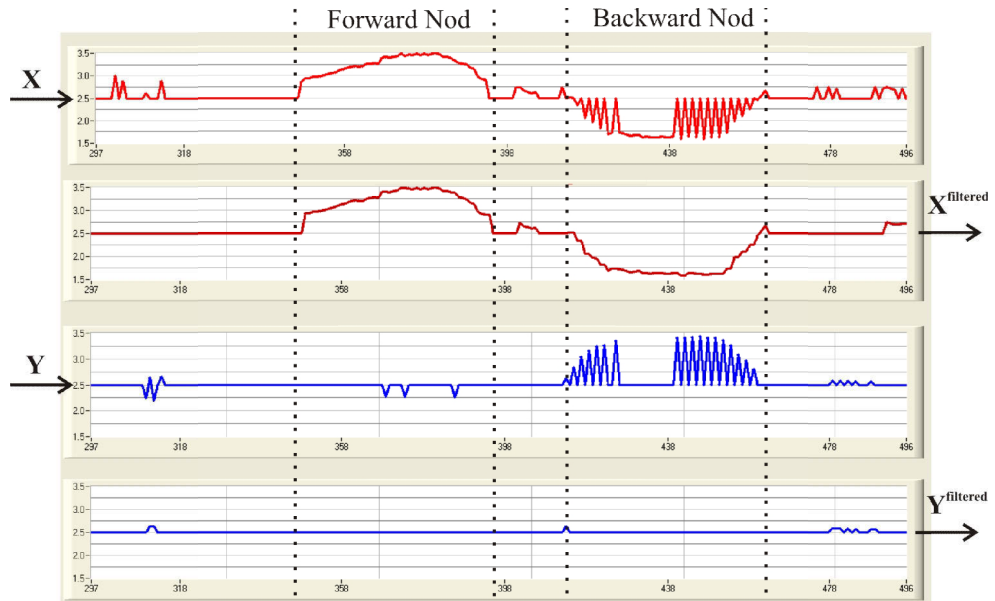


Figure 5.3: Head movement sensor outputs in response to forward nod and backward nod

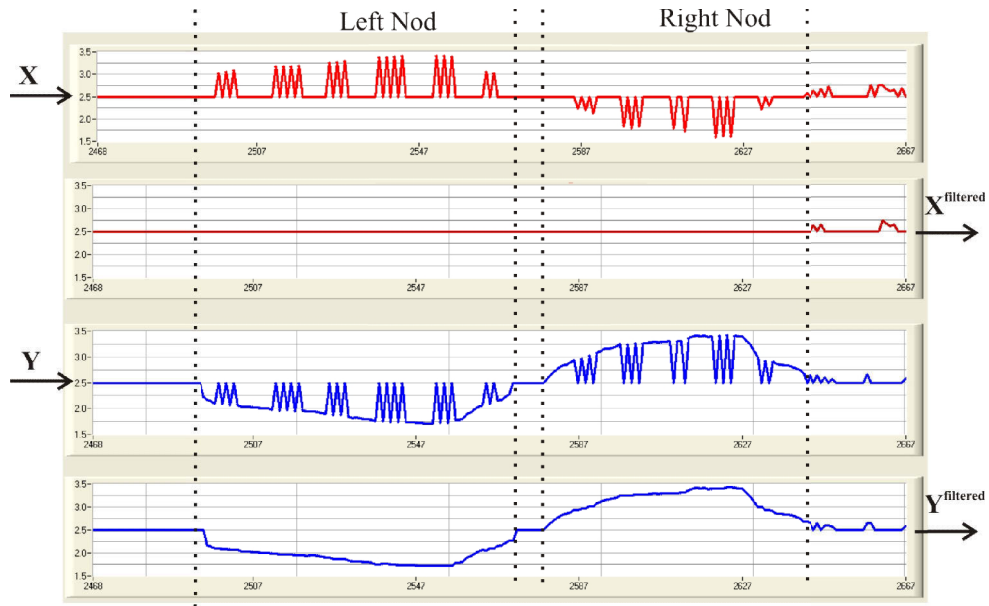


Figure 5.4: Head movement sensor outputs in response to left nod and right nod

As shown in these figures, there is high frequency noise mixed with the head movement outputs, thus, a real-time low pass filter is applied in order to remove the noise. After eliminating the noise, clean signals are obtained as displayed in the figures 5.3 and 5.4.

The filtered signals are then used to control the wheelchair manually. Unfortunately, the people with severe injuries have very poor control of the musculature supporting the upper body and neck, and consequently, the threshold - based calculation is always difficult to estimate the exact control command corresponding to inconsistent signals created by the impaired users. In order to overcome the challenges, the artificial neural network has been applied to detect head movement (Nguyen, Nguyen & Taylor 2006; Yuwono, Handojoseno & Nguyen 2011). In this project, the Bayesian neural network developed in chapter 4 is designed to recognise the control command for the head movement interface.

As demonstrated in the figure 5.5, the neural network classifier is designed to process the head movement signals and produce a control command which is one of four pre-defined commands: forward, left, right and stop. In particular, the network has the following specifications:

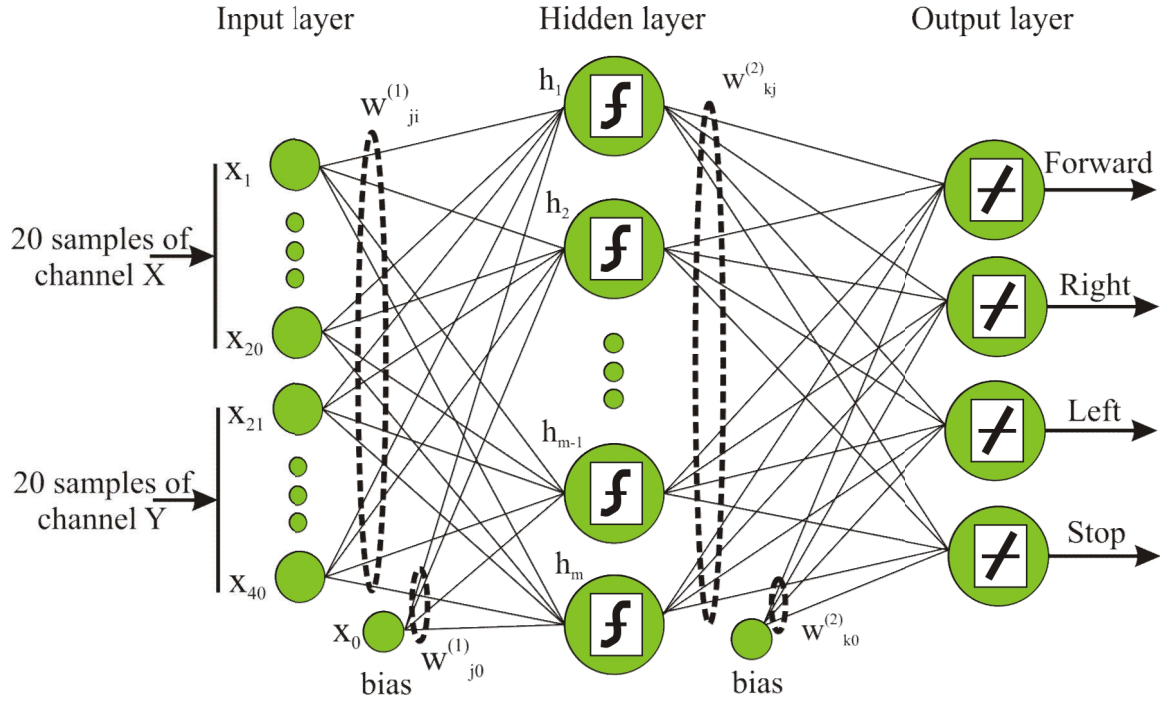


Figure 5.5: Feed-forward neural network for the head movement detection

- Input layer: 40 inputs which correspond to 20 samples of the channel X and 20 samples of the channel Y.
- Hidden layer: A number of hidden nodes are permitted to change from 1 to 8. The most appropriate structure will be determined during the training process for the head movement detection.
- Output layer: There are four outputs corresponding to the possible control commands: forward, right, left and stop.

The designed network is then trained under the supervision of the Bayesian framework developed in chapter 4 in order to find the most appropriate structure as well as the weight values for the head movement detection. After training, the network would be able to receive the signals from the head movement interface and produce the corresponding commands: forward, right, left or stop.

5.3.4 Shared Control Framework Implementation

The entire framework to share the control between the user and the machine for the SAM intelligent wheelchair is shown in the figure 5.6. At the beginning, the information about the environment and the user command have not yet been observed, therefore, it is necessary to initialize the posterior probabilities of the actions. The posterior probabilities across the defined actions F, R, L, S are set as follows.

$$\begin{cases} P(x_{t=0}^S) = 0.9 \\ P(x_{t=0}^F) = P(x_{t=0}^R) = P(x_{t=0}^L) = \frac{0.1}{3} \end{cases} \quad (5.21)$$

This initialization means that the SAM wheelchair would be stopped at the beginning, and the chance to have the mobile actions F, R, L is equal together.

Next, the environment information and the user command signals are updated in the real-time from the laser range finder URG-04LX and the head movement interface. The obtained data are then processed to find a number of free spaces as well as detect the user commands by means of the trained Bayesian neural network classifier.

From the results of the user command classifier and free space detection, the likelihood distributions would be modelled by the equations (5.12) and (5.16) as a consequence. The likelihood distributions then play a role of the inputs for the two consecutive update steps: the prediction and measurement update described in the equation (5.11). The result of this update process would be the posterior probabilities of a number of possible actions that the SAM wheelchair might take in the considering step. The decision about the optimal action is made based on the highest posterior probability $P(x_t|O_{1:t})$ that is likely to agree with both the user's intents and the environment information. Finally, the SAM wheelchair uses the trained autonomous neural network to provide the navigation assistance to perform the selected action.

In the next section, we will organise experiments to evaluate the developed shared control framework. The first experiment shows how the framework responds to the new data to

make the shared control decisions in the real - time operation. The second experiment is implemented to test the system with a number of different able - bodied users.

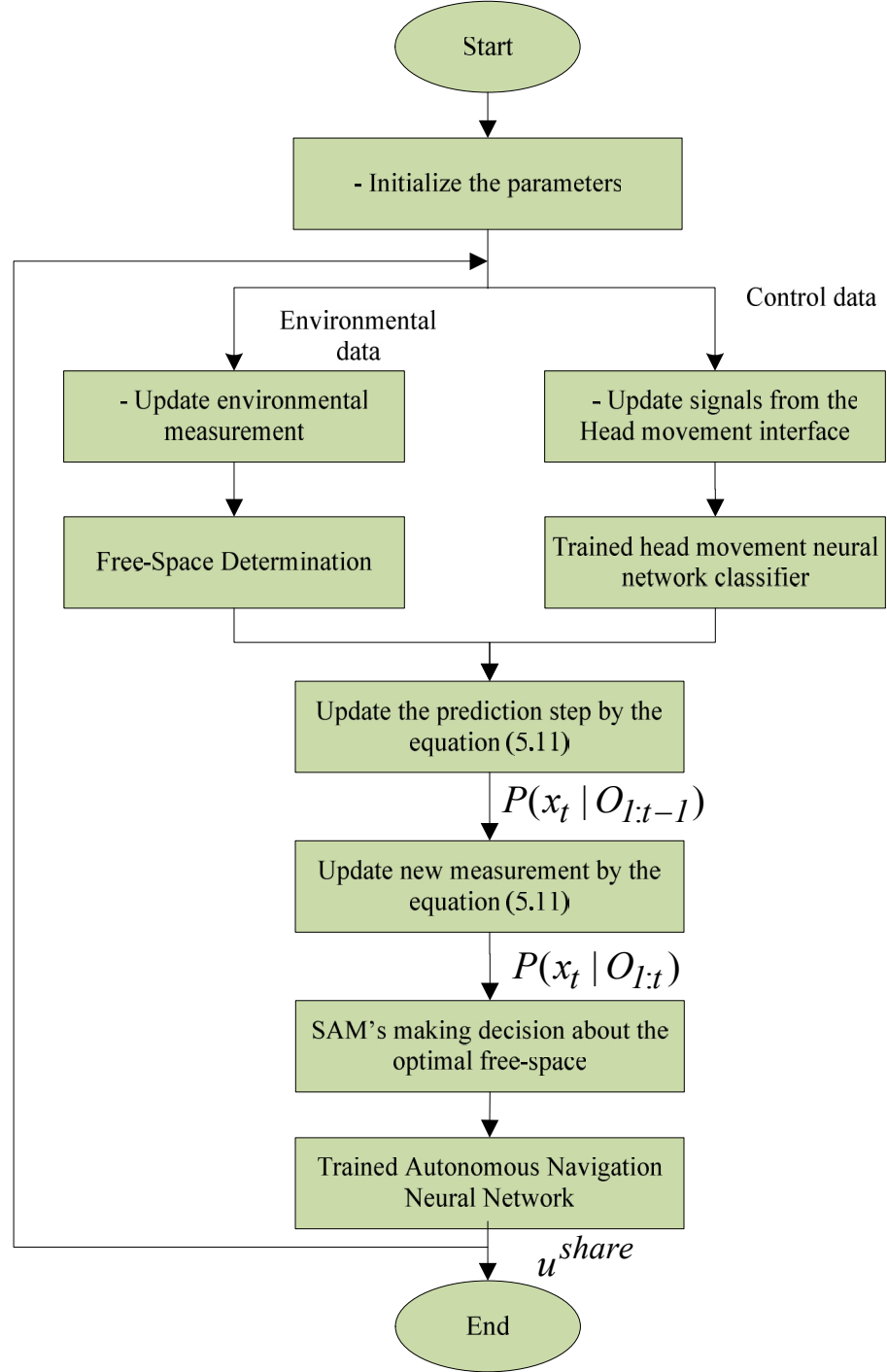


Figure 5.6: Bayesian Shared Control Framework for the SAM Intelligent Wheelchair

5.4 Experiment Results

5.4.1 Experiment 1: Advanced Bayesian Shared Control Strategy

The purpose of this real-time experiment is to demonstrate that the system using the Bayesian shared control framework is able to make the most appropriate action to navigate in the real-time. It also illustrates that the SAM intelligent wheelchair has the capability of gathering environmental information as well as detecting the user's command, then modelling and combining these information sources to justify its decisions.

Before the result of the proposed adaptive shared control strategy is shown, the head movement detection results using the Bayesian neural network classifier are presented first, the accuracy of this classifier will be the basis to update the prediction command probabilities within the shared control framework.

5.4.1.1 The Head Movement Detection

A Data acquisition

The protocol for the data acquisition was referenced from several previous works (Nguyen 2006; Nguyen, Nguyen & Taylor 2006; Yuwono, Handojoseno & Nguyen 2011). The Head movement training data was collected from six different healthy volunteers using the head accelerometer installed in the middle of a cap. 80 samples of command patterns for each person were taken as shown in the table 5.1.

Each command pattern consisted of 20 data points of the channel X and 20 data points of the channel Y as shown in the figure 5.7

User	Forward	Right	Left	Stop
1	20	20	20	20
2	20	20	20	20
3	20	20	20	20
4	20	20	20	20
5	20	20	20	20
6	20	20	20	20

Table 5.1: A number of command patterns of the five wheelchair users

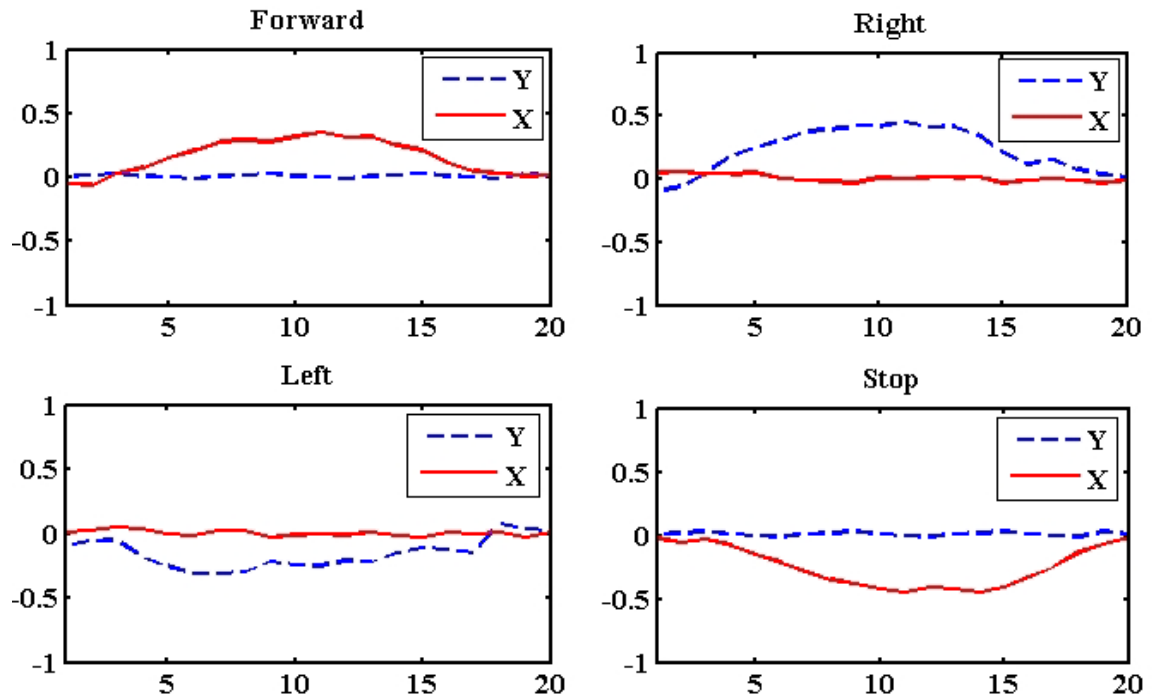


Figure 5.7: Command patterns of the first user

All samples were then divided up into two sets: training set and test set. The samples taken from users 1 to 4 were used to train the network under the supervision of the Bayesian framework; the performance of the trained networks was tested on the samples from users 5 and 6.

B Head movement detection results

During the training process, the structure of the designed head movement detection network was allowed to vary from 1 to 8 to select the optimal architecture. The figure 5.8 shows the evidence of the neural networks with different numbers of hidden nodes. The networks with three hidden nodes have the highest evidence. To consolidate the “best” structure for the head detection, the highest evidence networks with three hidden nodes and four hidden nodes were tested on the samples from the 5th user and 6th user.

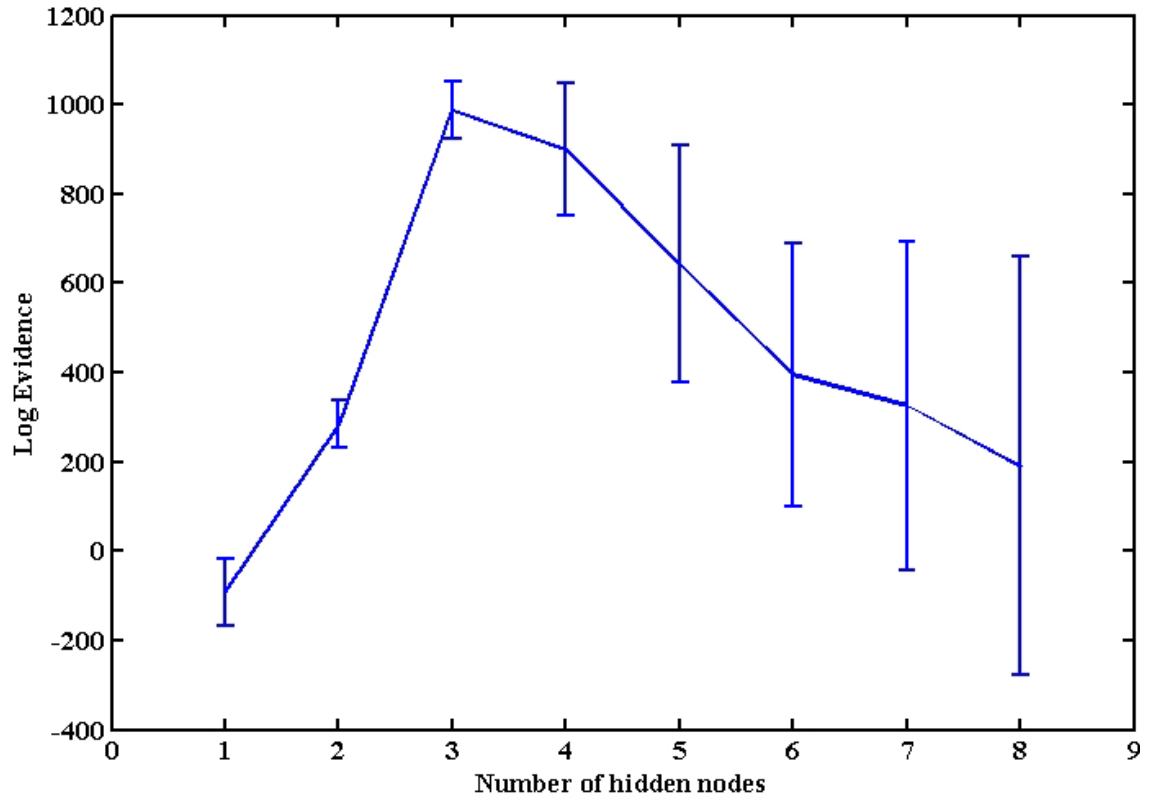


Figure 5.8: Log Evidence versus number of hidden nodes

The table 5.2 and 5.3 illustrate how accurately the trained networks can classify all of the movements.

		Predicted Classification			
	Movement	Forward	Right	Left	Stop
Actual Classification	Forward	35	2	2	1
	Right	2	36	0	2
	Left	1	1	37	1
	Stop	0	1	1	38
	Accuracy (%)	91.25			

Table 5.2: Confusion matrix of the trained network with 3 hidden nodes

		Predicted Classification			
	Movement	Forward	Right	Left	Stop
Actual Classification	Forward	34	2	4	0
	Right	1	36	0	3
	Left	6	0	32	2
	Stop	0	2	3	35
	Accuracy (%)	85.625			

Table 5.3: Confusion matrix of the trained network with 4 hidden nodes

It is obvious that the trained network with three hidden nodes approximately produces a prediction with 91 % accuracy, more accurate than that with four hidden nodes. As a consequence, the network with 3 hidden nodes is chosen to predict the user command for the following experiments. Based on the accuracy of the classifier, it could be said that if the output of the network predicts the command k , then the probabilities of the user commands can be presented by

$$\begin{cases} P(u_t^k) = 0.91 \\ P(u_t^{j \neq k}) = 0.03 \end{cases} \quad (5.22)$$

5.4.1.2 The SAM Performance with the adaptive shared control strategy

A) The wheelchair trajectory

After initialized and loaded with the necessary parameters, the SAM intelligent wheelchair was recruited to implement a real time experiment to test the proposed shared control method. The experiment was conducted in our laboratory's room located on the level 18 of the Building 1, University of Technology, Sydney where a number of free spaces were available. The head movement interface was connected to the system in order to command the wheelchair. The sampling time for collecting data from the environment and the user interface was set to be 100ms.

The figure 5.9 shows the environment topology and a trajectory produced by the wheelchair in the experiment. Starting from the point A, although asked to go forward, turn left, turn left, and stop at the point A, B, C, E respectively via the head movement interface, the wheelchair created a trajectory marked as the red dotted line. This trajectory is the result of the combination of both environmental information and user commands during the navigation process.

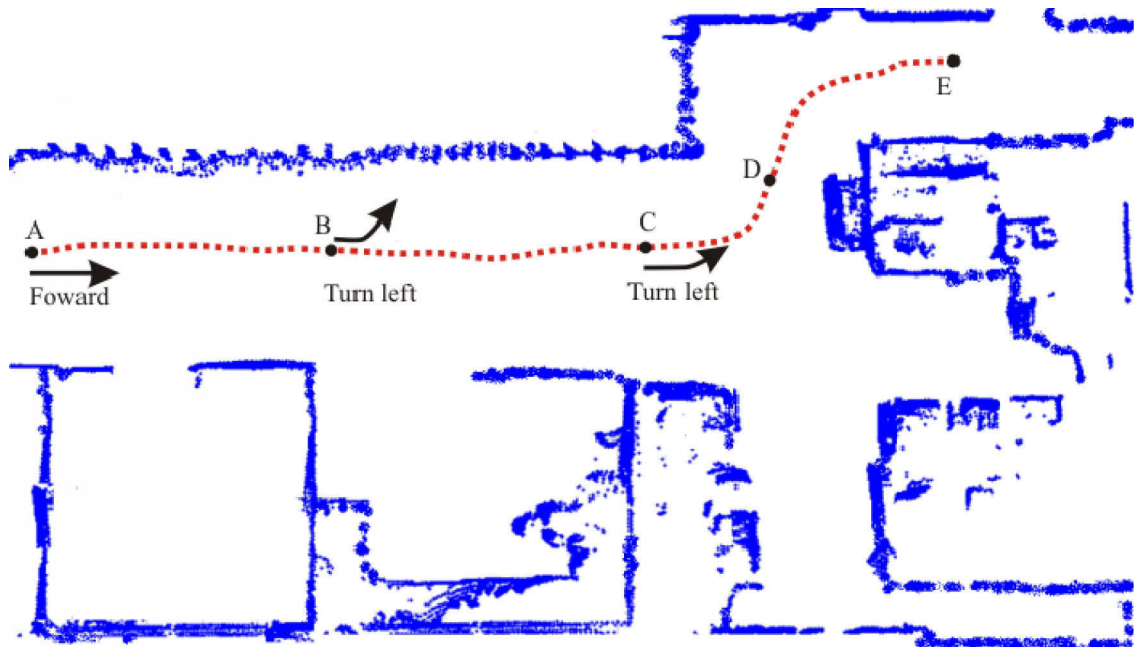


Figure 5.9: The Wheelchair trajectory when using the Bayesian Recursive Technique

B) Explanation

At point A, as shown in the figure 5.10, by intuition, the wheelchair might go forward or turn right into the door but should not turn left due to the glass wall; thus if forward

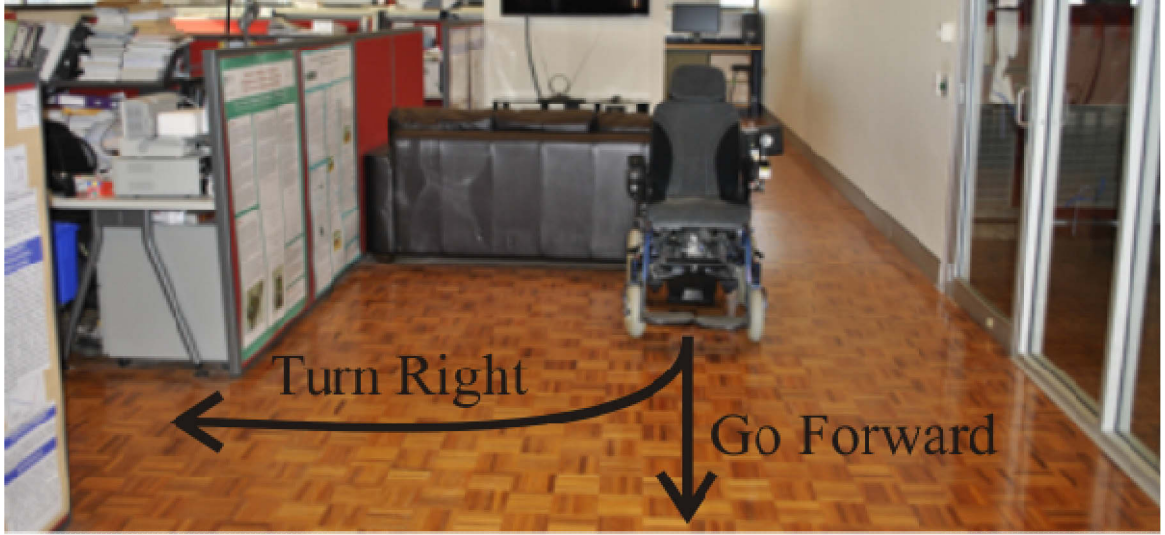


Figure 5.10: The scene and real-time program interface around point A

command is given at this point, it should go forward. And in fact, the wheelchair's response was exact as we expected. The figure 5.11 provides the probability values updated from the real-time measurement data in the prediction step and the measurement update step. Starting from the initial posterior probabilities $P(x_{t=0})$ by the equation (5.22), the prediction probabilities and measurement probabilities were updated by the equation (5.11). It can be seen that the wheelchair SAM remained in the stop position until the 5th time step because the posterior probability of the action S was the highest. At the 6th time step, the F command from the head movement interface was detected, leading to the increase of the probability of the F action, therefore, the SAM wheelchair started to move as the $P(x_6^F)$ was the highest. The probability $P(x_6^F)$ also caused the prediction probabilities to change at the next step $t=7$ accordingly, and finally, consolidated the $P(x_t^F)$ in the next steps as displayed in the figure 5.11.

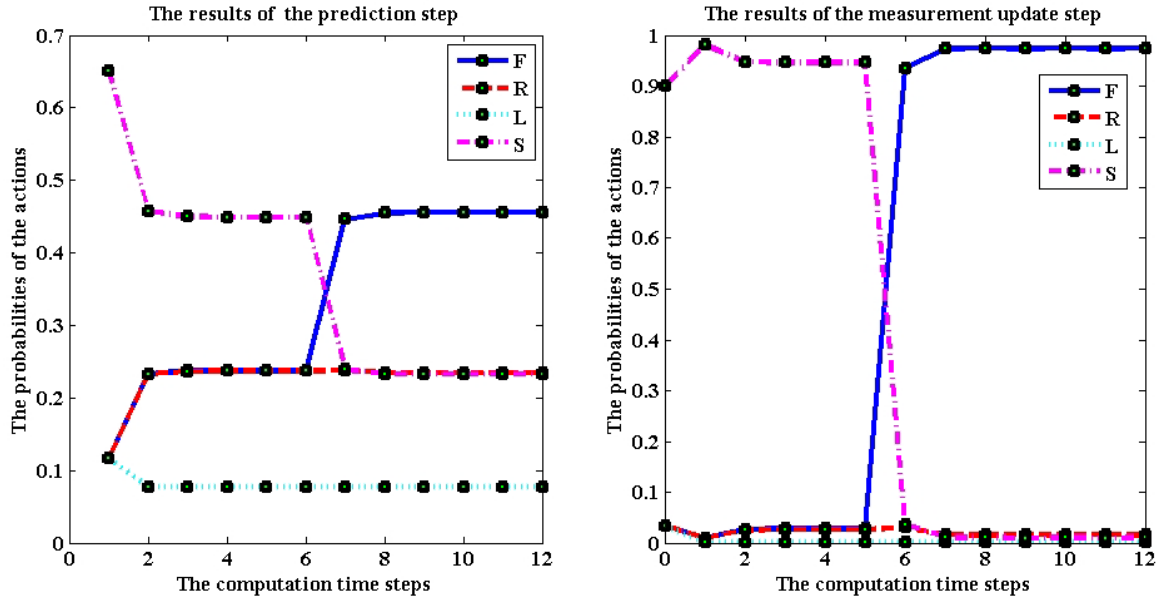


Figure 5.11: The evidence values at the point A when SAM started

At point B, when the user tested the system behaviour by the turn left command at $t=27$ although there was a wall on this side as the figure 5.12, the wheelchair still kept moving forward consistently. This is explained by the result of the update probabilities in the figure 5.13. The predicted probabilities across actions were unchanged at $t=27$ because they were based on the previous measurement at $t=26$. However, the posterior $P(x_{27}^F)$ at this time step was noticeably decreased as a result of the effect of the L command that was detected in the two time steps $t=27$ and 28 , then it gained its high value when the head movement stopped generating the L command. It is interesting that the probability of turning left action always remained very small although the action was supported by the user command measurement. This proves that the shared control method is able to prevent the system from collisions caused by incorrect commands issued by accident. In addition, despite being downgraded from the initial value, the probability $P(x_t^F)$ remained the highest value, and the SAM wheelchair still moved forward consistently. This suggests that the system using the proposed Bayesian shared control strategy avoids frequent action changes as the noisy factors affect it.



Figure 5.12: The scene and real-time program interface around point B

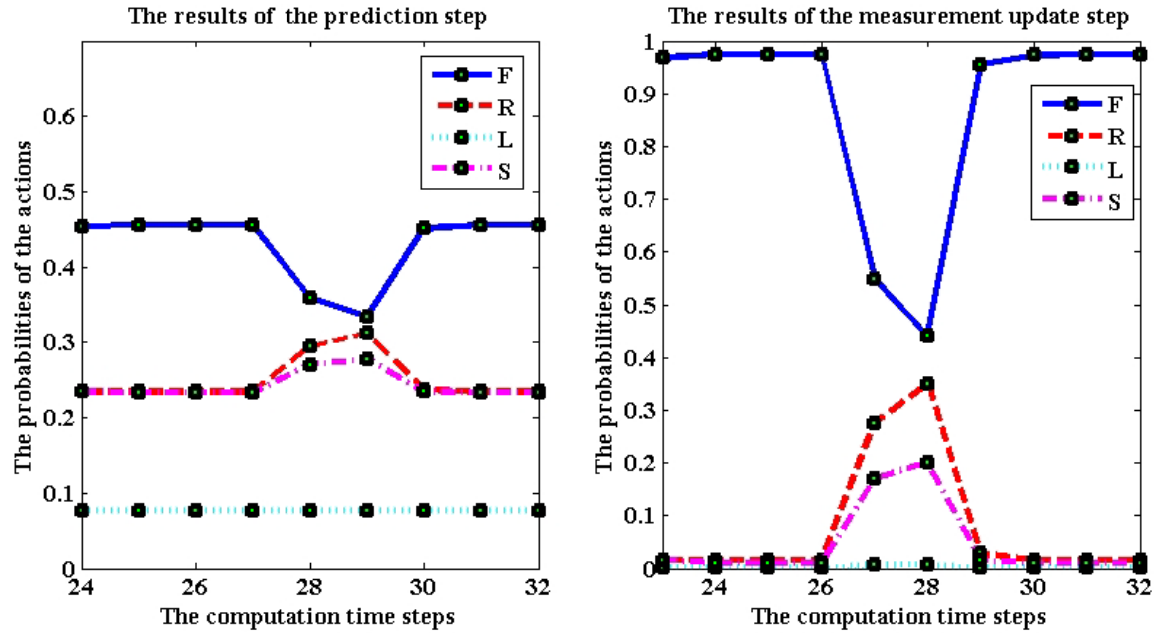


Figure 5.13: The action probabilities at the point B

Similarly, as the SAM wheelchair was approaching the point C, the environmental information supported all the actions F, R, L or S because there were three free spaces. Thus, the posterior probabilities mainly depended on the user commands. As shown in the figure 5.14, the SAM wheelchair started to turn left as the L command was detected at the computation time step $t=52$.

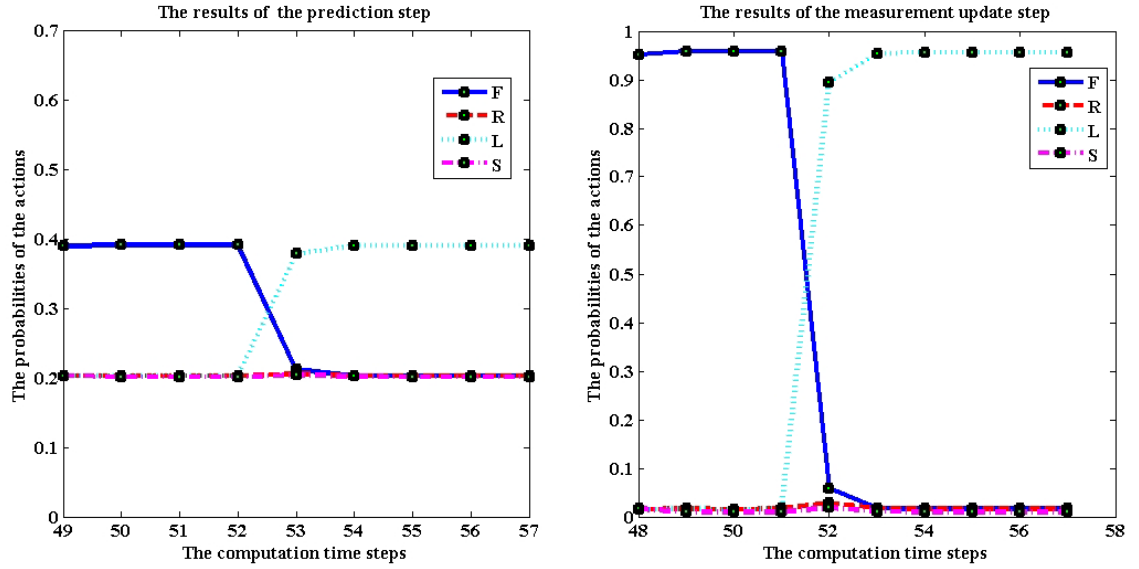


Figure 5.14: The action probabilities at the point C

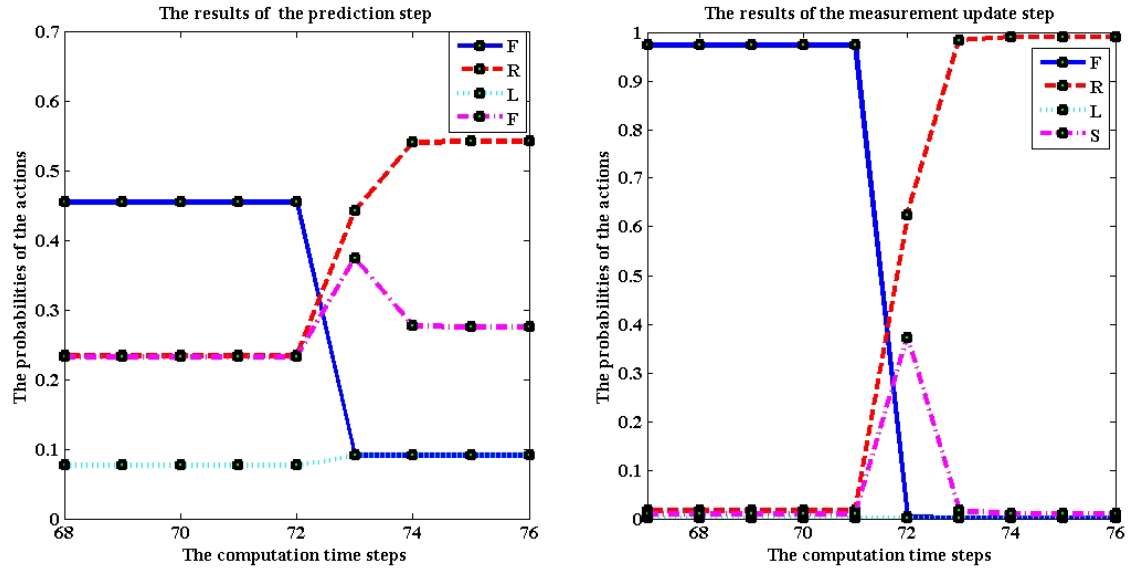


Figure 5.15: The action probabilities at the point D

The turning left action proceeded until the selected free space appeared in front of the wheelchair, and the SAM wheelchair automatically moved forward to enter it. At point D, because a free space on the right side only was detected, the probability of the turn right action became highest, therefore causing the system to turn right at $t = 72$ as shown in the

figure 5.15. We can see that the $P(x_{72}^L)$ was greater than $P(x_{72}^S)$ because $P(w_{72}^S | x_{72}^S)$ was set to be 0.6 while the free-space probability on the right side was estimated to be 0.9984. The situation would be different if the passage way was so narrow, with the probability of the free-space being less than 0.6, that then the SAM wheelchair would stop. Consequently, in order to make the system pass this narrow corridor, the user must provide a turn right command to increase the confidence for the system.

Finally, as the user tilted backwards to generate the S command, the action state of the system was changed at $t = 97$ from the F action to S action as the figure 5.16

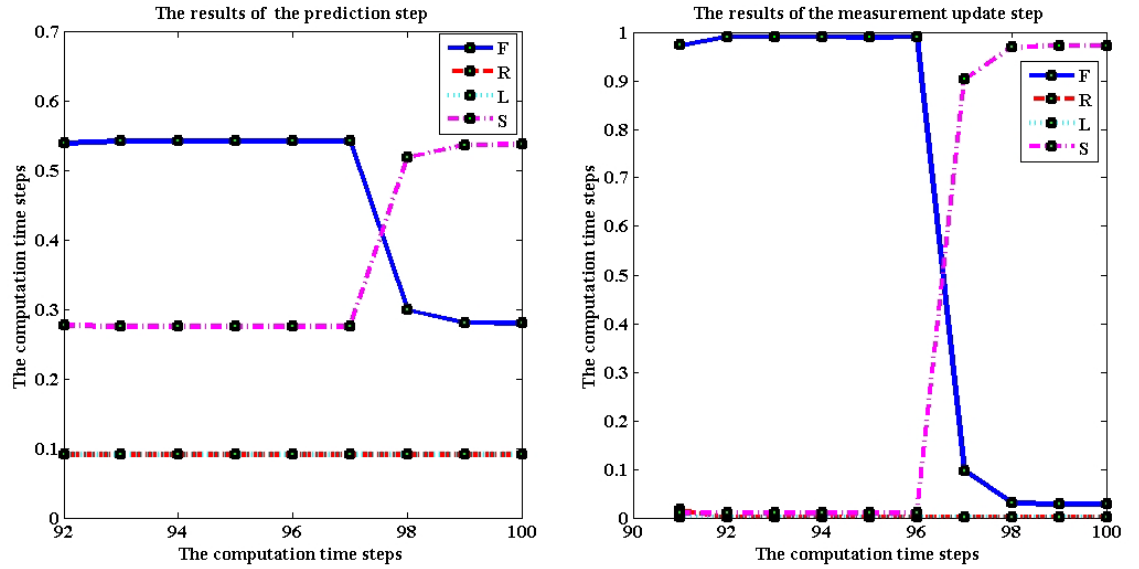


Figure 5.16: The action probabilities at the point E

5.4.2 Experiment 2: Evaluation of the SAM intelligent wheelchair with a group of able - bodied individuals

This experiment was conducted in corridors within the University of Technology, Sydney (UTS) in order to evaluate the performance of the SAM intelligent wheelchair. A group of 10 able - bodied participants were recruited to implement the real – time experiments by

using the head movement interface (HMI) to control the wheelchair. The main purposes of the experiments are to test the user's ability to control the wheelchair with the hands-free technology, and more importantly, assess the SAM wheelchair's ability to carry out the user's intentions using the Bayesian shared control developed in the section 5.3.

5.4.2.1 Experimental Study

A total of 10 able-bodied individuals with ages ranging from 20 to 35 years old completed the experiments on the SAM wheelchair controlled by the HMI. All the participants had no prior experience using wheelchairs.

Each user was asked to control the wheelchair by using HMI with and without any navigation support from SAM. In case only HMI is used, the system directly sends control signals as depicted in figures 5.3 and 5.4 to the motor controller. In the case of using HMI +SAM being used, the system implemented the shared control between the participants and the machine by the proposed shared control strategy as discussed in sections 5.2 and 5.3. The user controlled the wheelchair by tilting his head to imply his intentions, while SAM processed the intention of the user and combined this with environmental information to generate the control signals sent to the wheel motors.

The study was carried out in two sections, a training session and a testing session:

Training session: This session took place at level 4 of building 1, University of Technology, Sydney where they could practice driving the wheelchair. This place was very spacious, however, because of it being the main hall of the university involving many moving people during regular working hours, its space constantly varied; this environment was considered to be completely unknown for the wheelchair in each operation time. For this phase, each participant spent approximately 10 minutes to use HMI to drive the wheelchair with and without the assistance from SAM. Then, the same amount of time was spent to practice small tasks, for example, the user was required to control the chair to reach specified targets in the course. Throughout this session, the users learned the use of the interface as well as the underlying mechanism of the adaptability of the system.

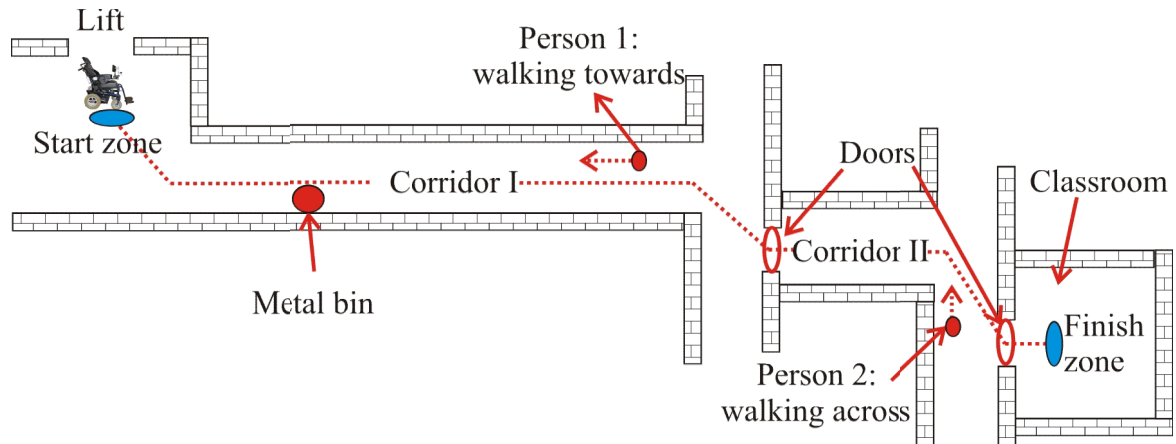


Figure 5.17: Designed course task for the evaluation experiment

Testing session: This session was carried out right after the training session and took place at level 7 of building 2, UTS. In this phase, the participant was asked to complete a designed task in the unknown and dynamic environment as shown in the figure 5.17. Following a visual and verbal description of the route, the subjects were asked to start from the lift, follow the long corridor I with just over 1m width, pass the front door of the second corridor, follow the corridor II, and finally enter the classroom. Each subject was asked to use HMI only and HMI+SAM in order to complete the route only one time for each control case. The obstacles that required avoidance in the test included walls, bins, corridor features and walking people.

5.4.2.2 Record Data and Data Analysis

Both quantitative and qualitative data were recorded throughout the experiments for evaluation by means of software in the onboard computer and the questionnaire. During experiments, all trials in the testing were also video recorded and then used to obtain the evaluation results described in the result section.

Quantitative measurement factors: A number of factors were measured throughout the testing session.

- *Number of successful tasks:* A task is considered successful if the wheelchair is able to reach the final destination. On the contrary, a task is considered to be a failure if the wheelchair fails to arrive at the target due to collisions, obstacle avoidance block or if the wheelchair malfunctions.
- *Time to complete the task:* This is the time taken to accomplish the task.
- *Number of the user – wheelchair interactions:* The number of interactions is simply defined as the number of times that the user purposely tilted his head in order to make noticeable changes in the wheelchair's state of motion.
- *Distance travelled:* The path length travelled to accomplish the task. This factor lets us know which case is the most efficient.
- *Number of collisions:* This number includes all collision times during the experiments. A collision is not considered a failure as long as the system is able to continue with a new command without any external interventions.

Qualitative factors: A questionnaire was made up to gather the subjective assessments from the participants. All questions in the questionnaire based on the NASA-TLX (Hart 2006; Noyes JM 2007) were related to comparing the ratings of the system operations on a scale of zero to five after the subject completed both training and testing sessions. The details about the evaluation questionnaire can be found in the appendix D. For the purpose of evaluating the task's independent workload associated with the use of the SAM intelligent wheelchair, the questionnaire was introduced to every participant at the beginning of the training course, and repeated during the training exercises as well as the testing session. After completing the testing session, each participant was asked to complete the answers to those questions.

Data analysis: To compare the performance of the users in these cases: HMI only and HMI+SAM, the pair-wise comparisons between the two control cases were performed using the standard t-test for the recorded data at the 1% significance level. This means that the test statement is accepted if it only has a less than 1% chance of being an accident.

5.4.2.3 Results

The overall result of experiments showed that all subjects were able to carry out the navigation mission along the designed route successfully. Both the quantitative and the qualitative results were analysed and discussed in this section.

A. Performance Trajectories

All participants had successfully driven the wheelchair to complete the designed route in

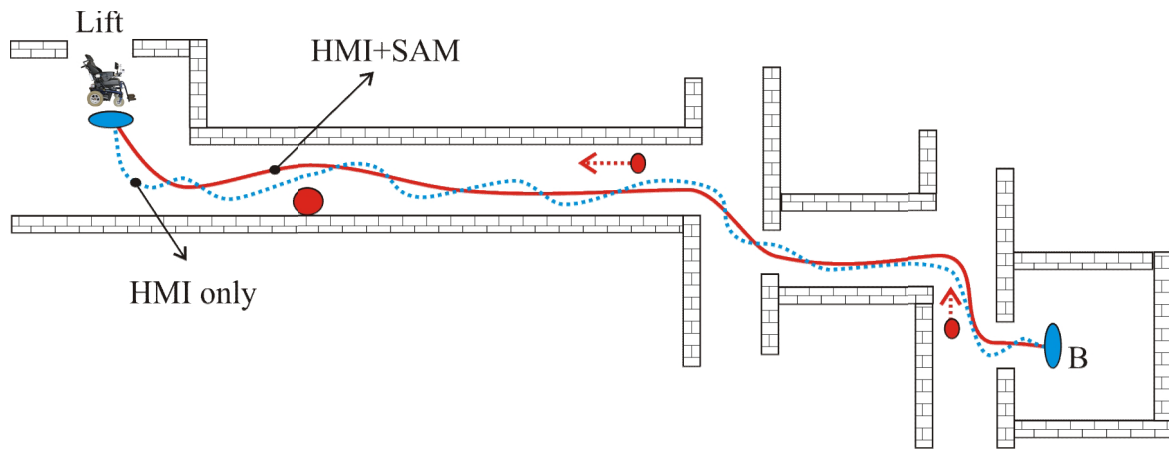


Figure 5.18a): Sample trajectories taken by the first user employing HMI only and HMI+SAM

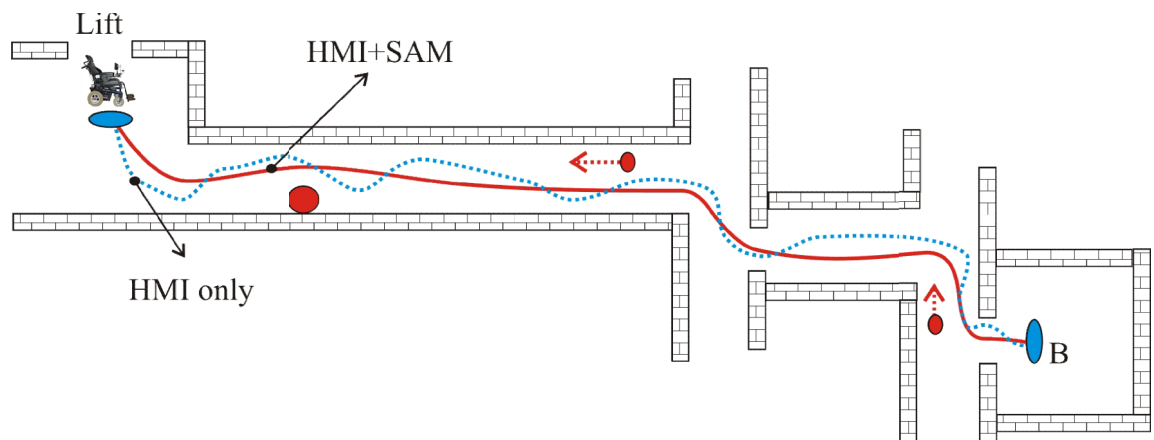


Figure 5.18b): Sample trajectories taken by the sixth user employing HMI only and HMI+SAM

two cases: HMI only and SAM+HMI. Sample trajectories created by the first participant and the sixth participant are depicted in the figure 5.18.

The HMI only: As shown in the figure 5.18 a), b), the blue dotted lines were the trajectories taken by the first and sixth users during the experiment. It is obvious that trajectories created by the two users as employing HMI were noticeably different. Across all the participants' trajectories, although they were different, the common feature of those trajectories was oscillation. The oscillated trajectories occurred because the operators had difficulty in changing their head movements corresponding to the changes of the environment. Sometimes, a user tilted or turned their head too much on one side, and then abruptly turned their head to the other side in order to prevent the wheelchair from colliding with the obstacles nearby, especially when the operator tried to avoid obstacles such as a metal bin or moving person.

The HMI+SAM: Trajectories created by this control case were slightly different across the participants. For instance, the red solid lines in the figure 5.18 a), b) show samples created by two of the participants. In comparison, it could be clearly seen that the trajectories using HMI+SAM were much smoother and more optimal than those with HMI only. This is because the autonomous navigation neural network was able to recognise the obstacles and the environment type, and thus, deliberately controlled the wheelchair corresponding to the changes of the environment. Unlike the above case in which the users often felt panic when getting near the obstacles, the users with HMI+SAM were supported to avoid the obstacles from a distance. In addition, the experiment also showed that the participants had the capability to ask the wheelchair to follow their intentions and reach the desired classroom.

B. Quantitative Results

The quantitative results measured in the testing session are displayed in the table 5.4.

	HMI only	HMI+SAM
Success	1.00±0.00	1.00±0.00
Collisions	1.47±1.30	0.20±0.41
Distance(m)	36.18±3.41	32.52±1.15
Time(s)	95.69±18.68	66.74±3.05
Interactions	79.80±17.43	10.07±3.45

Table 5.4: The quantitative performance results (mean ± standard deviation)

The task success: All the participants succeeded to navigate along the designed path with the two cases, which was a good indicator of the device utility.

The number of collisions: The table 5.4 showed the mean ± standard deviation and the figure 5.19 a) displayed the box plot of the number of collisions across the two control cases in the testing experiment. The t-test showed that the control of SAM wheelchair by HMI only had significantly greater number of collisions than that by HMI+SAM with p-value of $p = 0.0018$.

Distance travelled: The table 5.4 showed the mean ± standard deviation and the figure 5.19 b) displayed the box plot of the distances travelled across the two cases in the testing experiment. The distance travelled by the system as controlled by HMI only was statistically longer than that controlled by HMI+SAM with $p = 0.0017$.

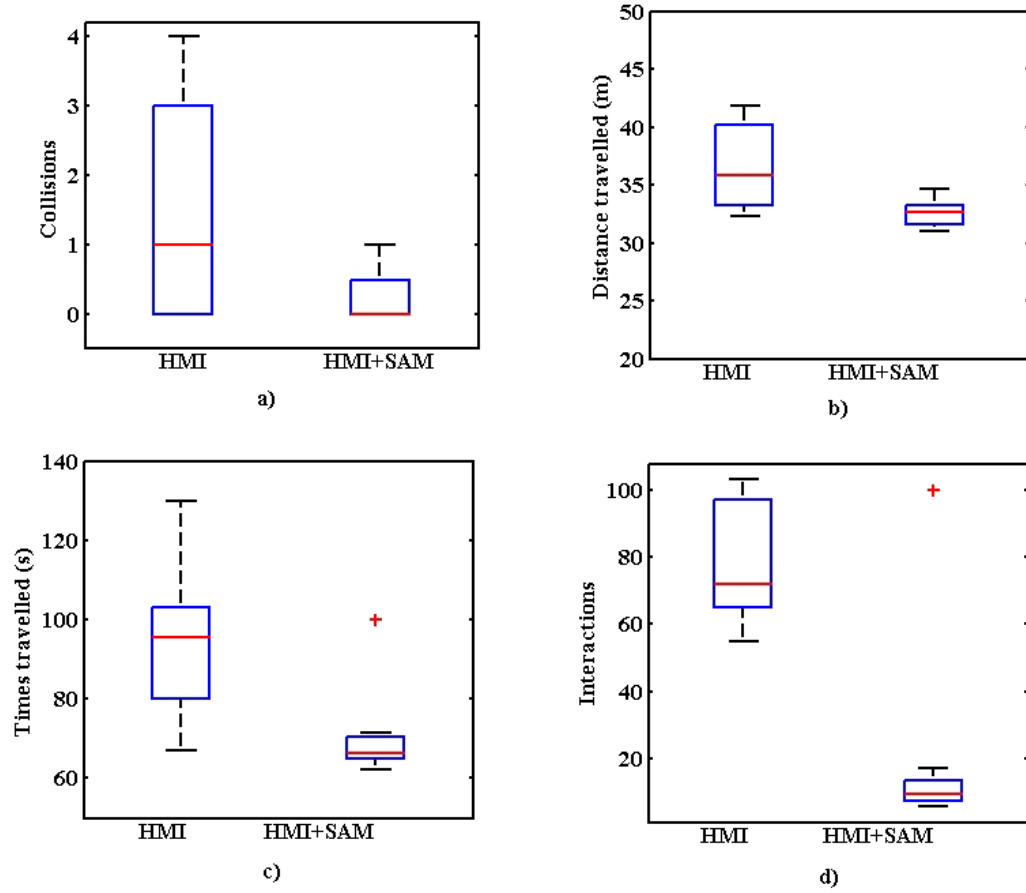


Figure 5.19: The box plot of quantitative performance results

Time to completion: The table 5.4 showed the mean $\pm$ standard deviation and the figure 5.19 c) displayed the box plot of the time to complete the testing task. The t - test showed that it took significantly longer when SAM was controlled by HMI than controlled by HMI+SAM with $p = 3.16 \times 10^{-5} < 0.001$.

Interactions between the user – wheelchair: The table 5.4 showed the mean $\pm$ standard deviation and the figure 5.19 d) displayed the number of the user – wheelchair interactions. The number of interactions that were necessary to drive the wheelchair by HMI only were significantly greater than those when the users were supported from the SAM intelligent wheelchair with $p = 2.26 \times 10^{-10} < 0.001$.

C. Qualitative Results

The results of the qualities measured in the testing session are displayed in the table 5.5.

	HMI only	HMI+SAM
Mental demand	2.80±0.94	2.27±0.80
Physical demand	4.27±0.70	1.80±0.86
Temporal demand	3.13±0.83	2.2±0.94
Performance	3.14±0.83	1.87±0.74
Effort	3.87±0.74	1.5±0.52
Frustration	3.20±0.77	1.46±0.52

Table 5.5: The qualitative performance results (mean ± standard deviation)

The mental demand: The table 5.5 showed the mean ± standard deviation and the figure 5.20 a) displayed the mental demand required to control the wheelchair with HMI only and HMI+SAM. The t-test showed that there was no significant mental demand difference between controlling the wheelchair by HMI only and HMI+SAM because of the p-value of $p = 0.072 > 0.01$.

The physical demand: The table 5.5 showed the mean ± standard deviation and the figure 5.20 b) displayed the physical demand. The t-test showed that the control of SAM wheelchair by HMI only required significantly greater physical demand than by HMI+SAM with p-value of $p = 3.34 \times 10^{-7}$.

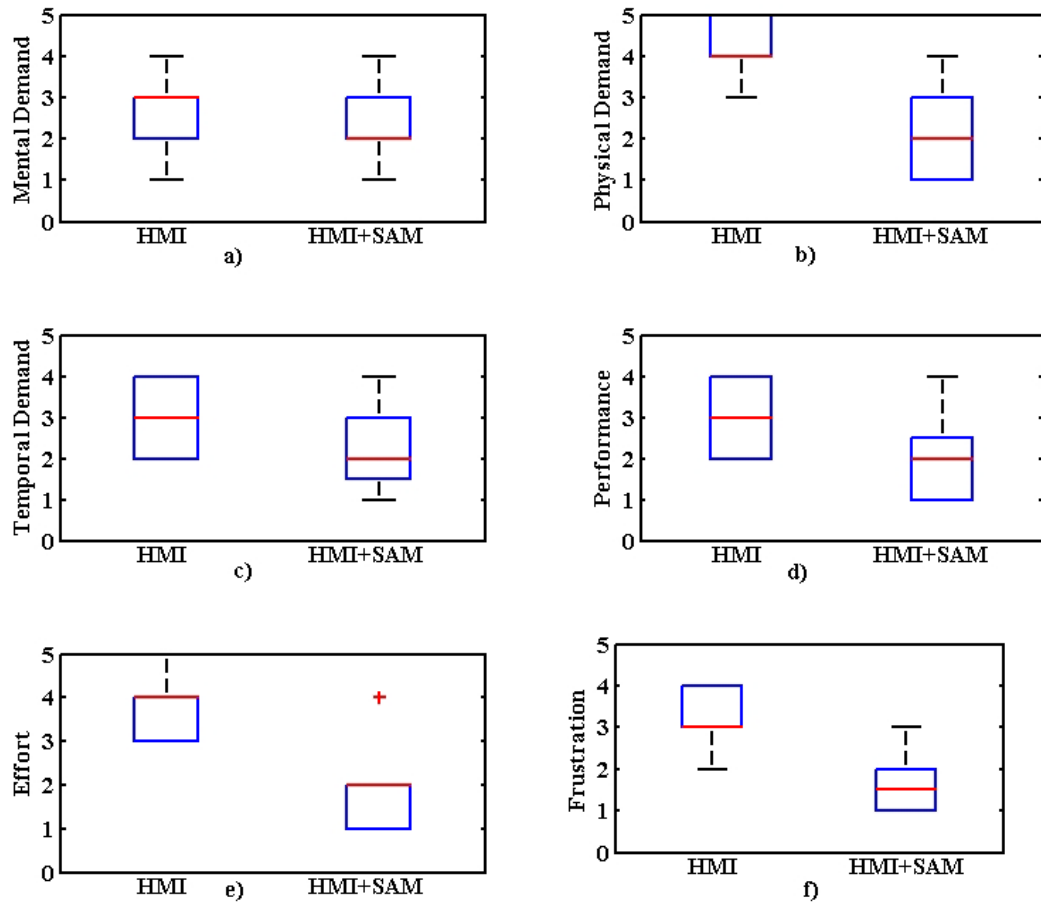


Figure 5.20: The box plot of qualitative performance results

The temporal demand: The table 5.5 showed the mean $\pm$ standard deviation and the figure 5.20 c) displayed the mental demand required to control the wheelchair with HMI only and HMI+SAM. The t-test showed that the temporal demand did not significantly differ between controlling the wheelchair by HMI only and HMI+SAM because of the p-value of $p = 0.0104$, just above 0.01.

The performance: The table 5.5 showed the mean $\pm$ standard deviation and the figure 5.20 d) displayed the performance. The t-test showed that the users controlling the wheelchair by HMI+SAM felt more statistically successful in performing the tasks than controlling by HMI only with $p = 5.38 \times 10^{-4}$.

The effort: The table 5.5 showed the mean $\pm$ standard deviation and the figure 5.20 e) displayed the effort. The t-test showed that the users controlling the wheelchair by HMI+SAM found it less hard in accomplishing the tasks than controlling by HMI only with $p = 5.56 \times 10^{-7}$.

The frustration: The table 5.5 showed the mean $\pm$ standard deviation and the figure 5.20 f) displayed the level of the frustration. The t-test showed that the users controlling the wheelchair by HMI+SAM felt more statistically secure in performing the tasks than controlling by HMI only with $p = 5.56 \times 10^{-7}$.

5.5 Discussion

The results of the first experiment, regarding the advanced Bayesian shared control strategy, indicate that the proposed method provides an effective way for designing the assistive intelligent wheelchair system. More importantly the method makes the correct and optimum shared control actions on the basis of combining the environmental information provided by the laser range finder URG-04LX, and the predicted commands of the user via the head movement interface. The selected action, corresponding to the highest posterior probability, is not only consistent with the user's intent, but also agreeable to the free spaces in the surroundings.

Unlike many other threshold - based share control techniques creating a moving fluctuation due to the noise factor, this method limits the frequency with which the system can change the actions caused by the driving limitation of the user. It is obvious that frequent navigation action changes can lead to an uncomfortable ride for the user and are thus explicitly limited. The first experiment shows that the SAM intelligent wheelchair avoids frequent action changes as the mistake commands are given by accident. The fact that users with disabilities produce the wrong commands by accident is very common, therefore, the proposed shared control method improves the navigation stability for the intelligent wheelchair system.

Another advantage is that the proposed method is made to operate in the real – time and unknown environment. Many shared control techniques require a topology map of the operating environment to estimate the user’s intentions; this means those systems usually only operate in the known environment and involve a lot of time – consuming computation steps, leading to the difficulties when operating in real-time. In the experiments, the proposed adaptive shared control mechanism did not interfere with the normal operation of the wheelchair in any trial. This implies that the SAM was able to devote most of its computation resources to providing the navigation assistance to the driver rather than making the adaptation decisions.

The second experiment shows the comparison of controlling the wheelchair by using HMI only and using HMI+SAM on both qualitative and quantitative measures recorded during the field trials. All participants used the head movement interface to control the wheelchair in the both cases. In terms of the quantitative, it can be concluded that participants performed significantly better when receiving navigation assistance from the semi-autonomous system across all evaluation criteria. (HMI +SAM) not only reduced the number of collisions, but also navigated in the more optimum trajectories. This is expressed via the distance travelled and the time completion. The wheelchair took less time to complete the navigation task and travelled shorter distances when the shared control between the user and machine was used to support the operators. Finally, the number of interactions as intuitive evidences shows that the users with HMI only required much more driving effort to complete the same navigation task than when controlled with HMI+SAM.

In terms of qualitative, participants did not experience differences in the mental demand and the temporal demand in both control cases, and those demands were relatively low. This shows that the experimental condition was simple enough for all participants. However, across other qualitative criteria, they felt significant differences between using HMI only and using HMI+SAM. Physical demand was significantly higher when using HMI only. This resulted from the fact that the operators were required to combine continuous control effort plus scanning of the surrounding obstacles to avoid. When using HMI+SAM, they were supported to avoid obstacles from the machine, and only required to command the SAM to change directions to complete the task with very low physical

demand. The second experiment also proved that the operators were noticeably satisfied with their performance, and felt much easier to complete the designed task when receiving the assistive navigation support from the SAM intelligent wheelchair. Without navigation support, they became tired and frustrated because they had difficulty determining precise control effort to keep the wheelchair away from obstacles in the narrow space like the experimental task. The low level of the frustration with using HMI+SAM also contributed from the fact that most of the participants felt comfortable in controlling the system; they were able to change the system's direction to obtain the navigation goal.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis reported the development of a semi-autonomous navigation system for people with impairments. Thanks to its autonomy in the navigation process and to the adaptive shared control intelligence, the SAM wheelchair enables disabled people to retrieve their personal mobile activities without the help of others.

As the assistive vehicle to the human users, developing the system that satisfies the strict requirements about the versatility and psychological acceptability is very challenging. Many autonomous algorithms which originated from the robotic field fail to successfully apply to intelligent wheelchair applications. For example, the intelligent wheelchairs using SLAM limit users to operating only in known environments. Classical obstacle avoidance methods such as VHF or VFF applied to the intelligent wheelchair create sudden changes in movement which are uncomfortable for the user.

Throughout this thesis, the real-time navigation system for the intelligent wheelchair is presented on the basis of combination of the Bayesian theory and artificial neural networks. The Bayesian theory contributes its advantages in dealing with uncertainty arising in the unknown environment as well as the limitation of the hardware devices. Meanwhile, the artificial neural network enables the system to learn human navigation skill via a number of training patterns. As a result, the trained autonomous behaviour of the chair has human characteristics rather than robotic characteristics typical of many other systems.

In the chapter 3, the SAM wheelchair has been introduced relying on the add-on design style. With this design, the system is easily converted from the standard electric wheelchair, or by removing the additional items can then revert to the normal power wheelchair without heavy modifications. The advantage of this design is to provide the flexibility in configuring the sensors, user interfaces, and it allows users to buy the system once, and then transfer it to multiple chairs over their lifetime.

Another main contribution of the chapter 3 focuses on building real-time maps and determining free spaces in the dynamic and unknown environment. A number of different sensors are required to continuously update data at the same time, and then the received data is combined in order to construct the real – time map. This method not only provides sufficient information for the system to navigate, but also improves the reliability in detecting surrounding obstacles. In addition, as the map contains the real – time information about surrounding obstacles, the SAM system can treat stationary objects and moving objects as the same when it navigates in the occupied environment.

Bayesian recursive technique is applied to provide an advance framework to determine free spaces for the SAM system. The free space is calculated based on taking account of measuring errors of the sensors and estimating the ability to navigate through that opening of the autonomous algorithm that the SAM system employs. The experimental results showed that the free space determination method makes the system robust with the limitation of the measuring devices as well as noise and unexpected changes of the operating environment.

The main contribution of the chapter 4 is to develop an effective autonomous navigation algorithm using Bayesian neural network. Unlike many traditional autonomous algorithms, we have proposed the technique based on the learning ability of an artificial neural network to create the system that not only safely and optimally navigates in the any environment types, but also produces the smooth trajectories suitable to the users with impairments. A number of navigate skill patterns collected from the data acquisition experiments have trained the network how to react with certain situations, and after training, the network is able to navigate the chair in the environment while avoiding surrounding obstacles. The

experiments showed that the technique successfully navigated the wheelchair in the dynamic and unknown environment with excellent performance qualities.

The performance of a neural network significantly depends on how the network has been trained. The maximum likelihood technique trains the network by minimising the error between the actual network outputs with the expected network outputs. However, this technique has the potential to make the training process over-fitting because of insufficient or bias training data set.

The chapter 4 described the method of using Bayesian theory to compare and rank various network architectures, and then, find the most probable weight values and structure of the network by evaluating the evidence of the networks based on the training data. The experimental results demonstrated that the network with four hidden nodes, having the highest evidence value, produces the best performance in the real – time navigation. This means that the Bayesian training framework is able to select the best model for the application.

The chapter 5 implemented an adaptive shared control strategy between the user and the machine. The fully autonomous system always results in the human operator being reluctant to use it because they feel they are being led by the machine. Conversely, the users of the fully manual system are incapable of controlling it due to physical or mental impairments. For these reasons, it is necessary to establish an adaptive shared control mechanism that allows the users to control the system with their own abilities while still providing the mobile support during the navigation.

The experimental results indicated that the proposed shared control strategy based on the Bayesian theory is capable of operating in the real time and provides an effect way for making correct and optimal shared control decisions on the basis of combining the environment information and the predicted commands of the user. Moreover, the method was also proven experimentally to improve the navigation stability by preventing the system from frequent action changes due to noises and the mistaken commands by the users.

Finally, a number of able-bodied people have been recruited to use the developed semi – autonomous navigation system for the evaluation purpose in terms of both qualitative and quantitative criteria. The experimental results were very positive. Statistically, all participants performed significantly better when receiving navigation assistance from the semi – autonomous system. This convinces us that our SAM system is a valid alternative for the people with disabilities, and that it might be integrated into next generation of navigation assistants.

6.2 Future Work

Besides significant improvements made for the intelligent wheelchair throughout this project, some further work could be done in order to improve the system performance and the robustness. Although semi-autonomous navigation algorithm is developed to operate in any unknown environment, the system is actually limited to operate in the indoor environment due to the limitation of the sensors that are not suitable for outdoor use. For instance, the laser range finder URG-04LX is well known not to work properly in sunlight conditions. Therefore, the sensor could be replaced by other sensors that are suitable for the outdoor use.

Another sensor – related problem requiring further work is that the laser range finder URG-04LX has limited detected range and has difficulties in sensing transparent surfaces such as glass walls, or plastic panes. Although the wheelchair has been instrumented with two sonar range finders, the problem is only less severe, but not completely solved. My recommendation is to also use 3-D camera sensors such as spherical vision cameras that are able to provide the image of all surrounding obstacles for the obstacle avoidance algorithm.

The semi-autonomous control software has been designed to work with different interfaces to accommodate people with various levels of disability. However, the system is only tested with able-bodied users. In order to be commercialized, the SAM wheelchair must be fully tested with disabled users in a hospital ward or elderly care centre for a certain period of

time. The testing results can then be used as an important information source to refine the semi-autonomous design.

Finally, our current design uses an onboard personal computer attached to the back of the chair. Therefore, it is necessary to convert the design into the embedded system. This is because the use of the general – purpose computer is far too costly for the commercial product. In addition, the embedded system would improve a number of performances such as the power consumption, size – limitations, reliability or real – time performance, etc. My suggestion is to use the BeagleBone board with just the credit card size to replace the mini – computer of the current intelligent wheelchair design. The BeagleBone development supported by Texas Instruments allows the user to access full computing power of the ARM Cortex A8 processor using the underlying Linux – based operating system. Moreover, a host of feature extension boards and a wide variety of open source libraries written for the BeagleBone make the design conversion easier and more reliable.

Appendix A

Power Wheelchairs

A.1 Overview of Power Wheelchairs

By definition, an electric powered wheelchair (EPW) is a wheelchair that is propelled by means of an electric motor rather than manual power. At first, an electric-power wheelchair was invented by George Klein to assist injured veterans after the World War II, then it has been popularly used by people for whom walking is difficult or impossible due to illness (physiological or physical), injury, or disability. Moreover, a power wheelchair is nowadays considered as an aid to mobility and a means of increasing independence and is not a sign of disability, as more users have transited from manual mobility to powered mobility.

To accommodate to a variety of users, a number different types of power wheelchairs are also available. However, in terms of power wheelchair frames, there are two basic configurations: the conventional frame and the power base. The conventional power wheelchair frame is the conventional frame which is very similar to a manual wheelchair frame but has batteries, motors and electronics mounted on it. This class of power wheelchair, shown in figure A.1, can be folding like the folding manual chairs or non-folding like the manual rigid chairs. Both the folding and non-folding conventional power wheelchairs have a couple of drawbacks. They never have electronics that can be upgraded, so if the user develops some added disability that prevents them from being able to operate

the chair, a new chair must be purchased. If the user has a dramatic weight gain or loss, the frame can't be adjusted or re-sized to fit properly without incurring a fair amount of cost.



Conventional folding chair



Conventional non-folding chair

Figure A.1: Power wheelchairs with conventional frame

Power base wheelchairs consist of two basic components, the base which includes the frame, wheels, batteries and electronics, and the chair which includes an adjustable seat frame, arm, footrests, cushioning and drive control. Almost every power base has electronics which can be upgraded in the event of reduced abilities and the seat frames which can be adjusted for changes in size and seating options. This adaptability of power bases means the chair will not have to be replaced until it is actually worn out rather than when the user changes condition. Power bases come in three wheel configurations, front wheel drive, mid wheel drive and rear wheel drive as shown in figure A.2. The configuration the user chooses is dependent on the type of terrain the user will usually encounter and the features important to the user. Each type of wheel configuration has different driving characteristics and often a user who is replacing a previous power wheelchair will have a bit of difficulty getting used to a different configuration.



Figure A.2: Power wheelchairs with power base

For the intelligent wheelchair project at the Centre for Health Technologies, University of Technology, Sydney, The Invacare Roller M1 with rear wheel drive type was selected for modification because it is the most conventional configuration and requires no learning by users who have experience with older models of chairs. This chair is very stable at high speeds although it has the largest turning radius making it a little challenging to use in small places.

A.2 Power Wheelchair Components

A.2.1 Battery

The battery is one of the major limiting factors in power wheelchair performance. Because of wheelchair characteristic use, manufacturers often use deep-cycle batteries for power wheelchairs that are expected to provide long-running power and to last charge after charge. Deep cycle batteries are constructed with thick grids of antimony lead alloy and a dense paste of active material to withstand discharge and recharge cycles. This construction allows the deep cycle battery to deliver sustained power with low current drain for extended periods of time.

There are several different types of deep-cycle batteries available for power wheelchairs: Wet (or flooded), Gel, Absorbed Glass Mat (AGM). Wet batteries use the chemical reaction

between lead and sulphuric acid to create electrical energy. Gel batteries contain a mixture of sulphuric acid, fumed silica, pure water, and phosphoric acid, which forms a thixotropic gel. AGM batteries have an absorbent glass mat sandwiched between the plates, saturated with acid electrolyte, but with none free to spill. This “sandwich” allows uniform distribution of the electrolyte over the plates, and reduces the chance of battery damage caused by vibration and jarring. Table A.1 summarizes some positive and negative aspects of these battery types.

Types	Positive aspects	Negative aspects
Wet (Flooded) batteries	- Cheaper. - Less vulnerable to overcharging. - Great performance with careful maintenance - Lighter per Ah compared to most Gel or AGM’s.	- Require maintenance. - Battery acid can leak, causing corrosion and damage to chair and wiring. - Not approved for airline travel. - High rate of self-discharge when left sitting (6-7% per month).
Gel batteries	- No maintenance. - Cannot leak. - Operate better than wet batteries in low temperatures. - Less gas released when charging than wet batteries. - Approved for air travel. - Longer life cycle than wet batteries	- Expensive. - More weight per Ah than wet batteries. - Susceptible to overcharging.
AGM batteries	- No maintenance. - Can’t spill or leak. - Shock resistant. - Minimal gasses released when charging. - Low self-discharge rate (3% per month at 77°F). - Approved for air travel.	- Highest cost. - Susceptible to overcharging. - New technology.

Table A.1: Properties of deep-cycle battery types

The life of a deep-cycle battery is directly affected by the depth of each discharge cycle. A discharged with 70% remaining capacity will last much longer than one discharged to only 20% remaining capacity. The ideal compromise between initial cost, battery life and

usefulness is to aim for approximately 50% discharge between recharges. In addition, another greatest cause of battery failure is when the battery is left in a discharged state for a prolonged period of time. If the battery is not recharged to full capacity quickly, it can lead to permanent loss of part of the battery's capacity. This is particularly the case with wet batteries. Also leaving the battery in a discharged condition will also increase the likelihood of failure from a short circuit.

A.2.2 Motors

Without a doubt, electric wheelchair technology has come a long way since George Johann Klein, a Canadian who worked for the National Research Council of Canada, invented the electric wheelchair in the 1950s. A key component of today's advanced wheelchair designs is an electric motor and its controller. The propulsion system of power wheelchairs typically consists of a pair of electric motors as shown in figure A.3, one for each drive



Figure A.3: Wheelchair gear-motors

wheel, and a drive train consisting of gears, belts and other mechanical elements, which couples the motor's shaft to the drive wheel shaft.

Most wheelchairs utilize two permanent magnet DC (PMDC) motors. Not only is the wheelchair's DC motors called upon to run smoothly and quietly and to help ensure overall satisfaction among electric wheelchair operators, but the motors have linear torque-speed profile making them easy to control. In the market, there are several different models of 12V and 24V motors that are particularly well suited for meeting the power needs of front-wheel drive, mid-wheel drive, and rear-wheel drive electric wheelchairs. Although the overall efficiency of PMDC motors under light loading is found to be 60-70%, under loads typical for power wheelchairs, these motors are about 45% efficient. A drop in motor efficiency increases the current drawn from the battery and decreases battery life and efficiency.

A.2.3 Electric Motor Drive

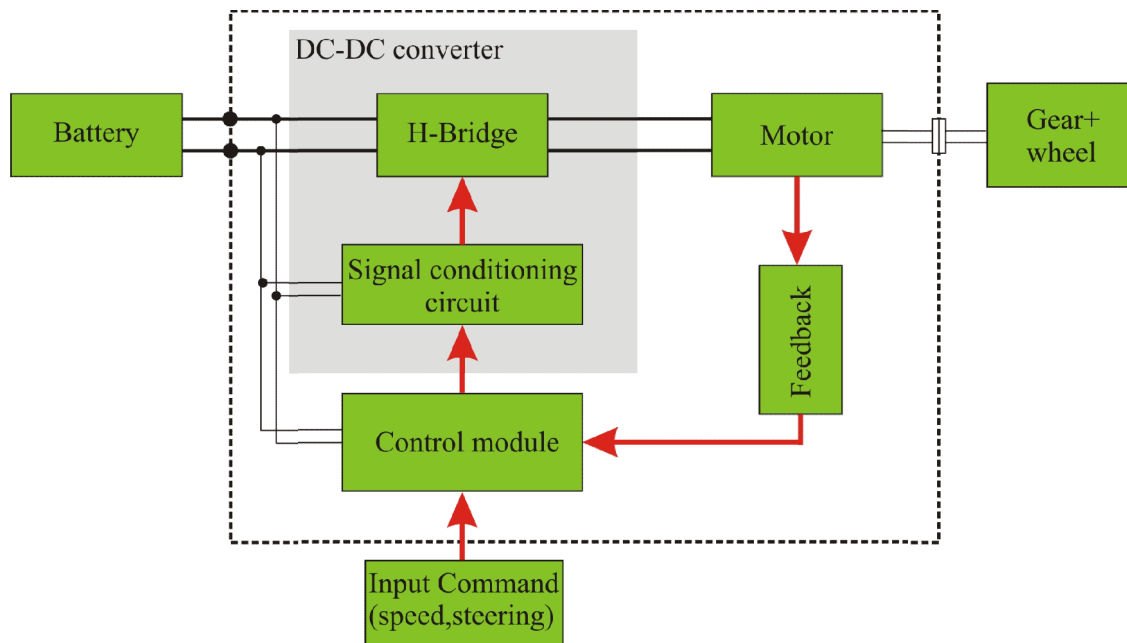


Figure A.4: Block Diagram of an Electric Motor Drive

A block diagram of an electric motor driver is shown in the figure A.4. The driver, also

called a DC-DC converter, controls each motor with a high frequency, square - wave, pulse – train that rapidly turns each motor on and off. Speed and torque generated by each motor are controlled by modulating the pulse width. Typically, the motor drive consists of a signal conditioning circuit and a H-Bridge circuit. The purpose of signal conditioning circuit is to interface the controller and the H-Bridge circuit, and the purpose of H-Bridge circuit is to vary the current in the windings of the PMBDC motor.

The H-bridge, named for its schematic appearance, is able to move current in either direction through the motor winding. To understand this, the H-bridge must be broken into its two sides, or half-bridges. Referring to figure A.5, Q1 and Q2 make up one half-bridge while Q3 and Q4 make up the other half-bridge. Each of these half-bridges is able to switch one side of the PMBDC motor to the potential of the supply voltage or ground. When Q1 is turned on and Q2 is off, for instance, the left side of the motor will be at the potential of the supply voltage. Turning on Q4 and leaving Q3 off will ground the opposite side of the motor. The arrow labeled IFWD shows the resulting current flow for this configuration.

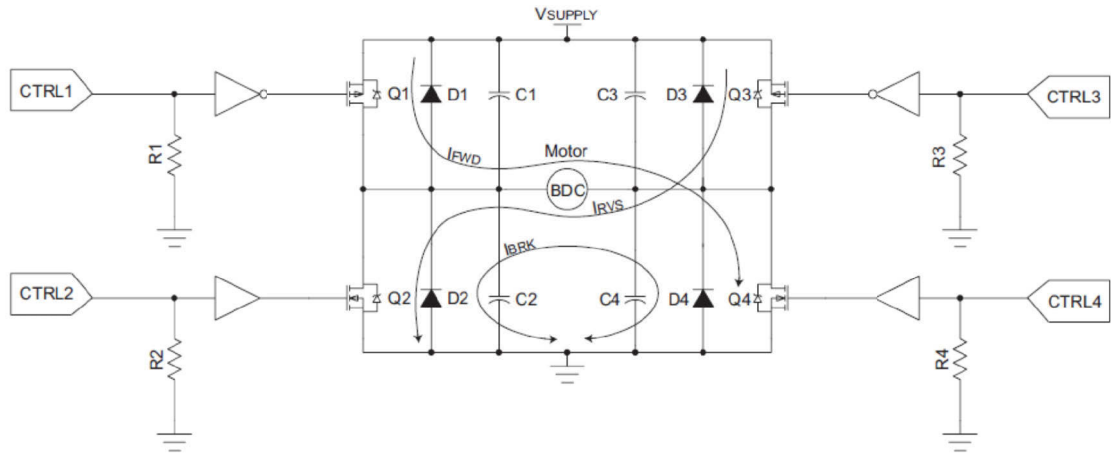


Figure A.5: Bi-direction PMBDC Motor H-Bridge Circuit

The different drive modes for an H-bridge circuit are shown in table A.2. In Forward mode and Reverse mode one side of the bridge is held at ground potential and the other side at V_{SUPPLY} . In figure A.5 the IFWD and IRVS arrows illustrate the current paths during the Forward and Reverse modes of operation. In Coast mode, the ends of the motor winding are left floating and the motor coasts to a stop. Brake mode is used to rapidly stop the PMBDC motor. In Brake mode, the ends of the motor are grounded. The motor behaves

	Q1 (CTRL1)	Q2 (CTRL2)	Q3 (CTRL3)	Q4 (CTRL4)
Forward	On	Off	Off	On
Reverse	Off	On	On	Off
Coast	Off	Off	Off	Off
Brake	Off	On	Off	On

Table A.2: H-Bridge modes of operation

as a generator when it is rotating. Shorting the leads of the motor acts as a load of infinite magnitude bringing the motor to a rapid halt. The IBRK arrow illustrates this.

There is one very important consideration that must be taken into account when designing an H-bridge circuit. All MOSFETs must be biased to off when the inputs to the circuit are unpredictable (like when the microcontroller is starting up). This will ensure that the MOSFETs on each half-bridge of the H-bridge will never be turned on at the same time. Turning MOSFETs on that are located on the same half-bridge will cause a short across the power supply, ultimately damaging the MOSFETs and rendering the circuit inoperable. Pull-down resistors at each of the MOSFET driver inputs will accomplish this functionality (for the configuration shown in figure A.5).

A.2.4 Control Module

As shown in the figure A.5, a control module is responsible for converting the signals from a user interface (joystick for instance) into the control signals for driving the wheelchair motors. The control module is microprocessor based and has many adjustable parameters.

Most wheelchair control systems have been designed with feedback for sensing whether the motor is responding properly to the control signals. Such control systems are able to adjust the motor torque so as to maintain near constant speed while the load varies in response to changes in the terrain (incline, bumps) and surface (linoleum, carpeting, concrete, grass and sand). Adjustable controller parameters include high/low speed range selection, maximum high speed, maximum low speed, maximum acceleration and turning speed, tremor dampening (low pass averaging of joystick position for example), maximum power delivered under high torque, and energy saver (reduces speed and increases range when the

battery is low).

At the Centre for Health Technologies – University of Technology, Sydney, the wheelchair Roller M1's control module is joystick DX master. Parameters store in DX master can be configured in three methods: Hand held programmer (HPP), Wizard 5 and HPP plug – in.

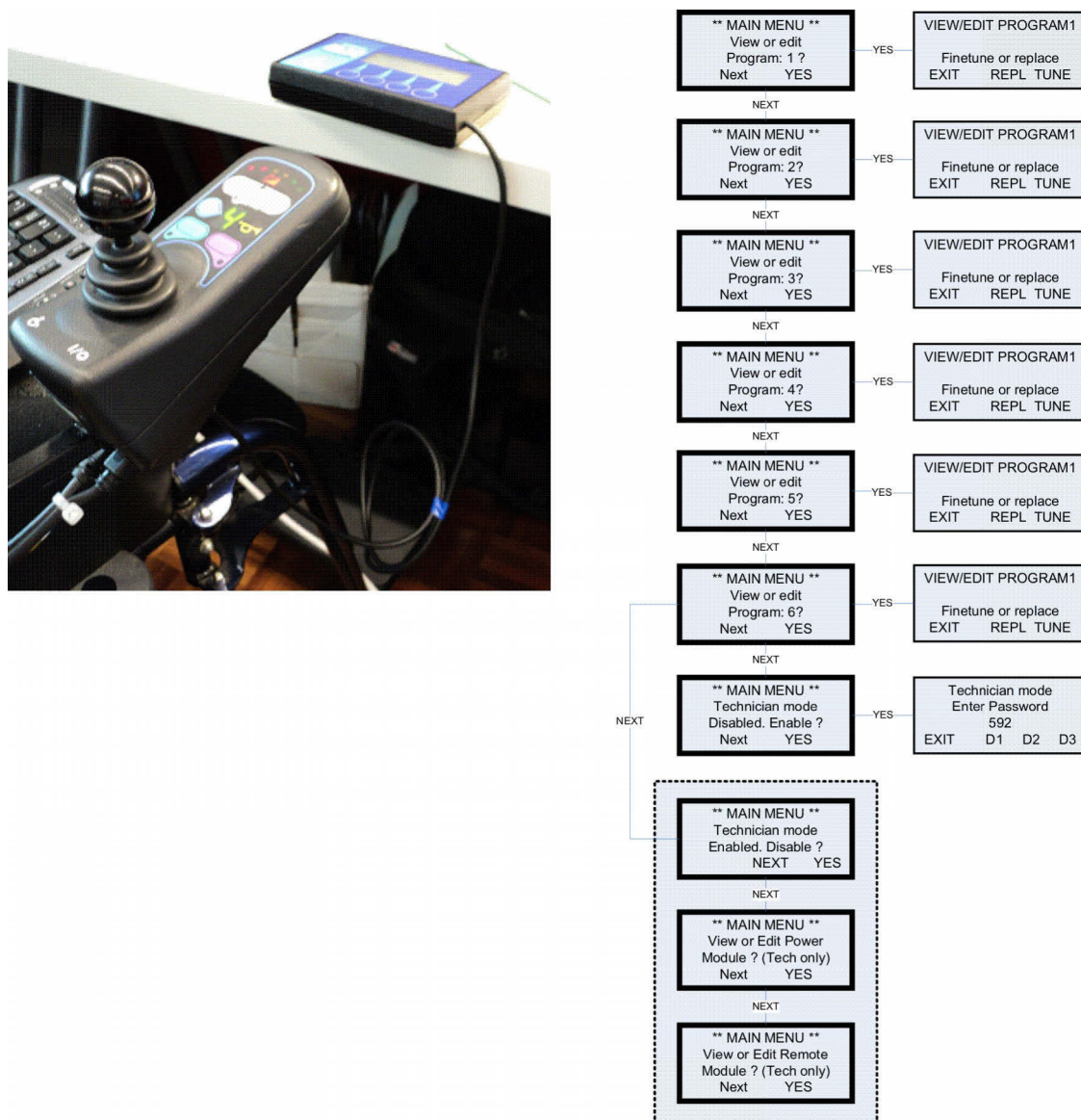
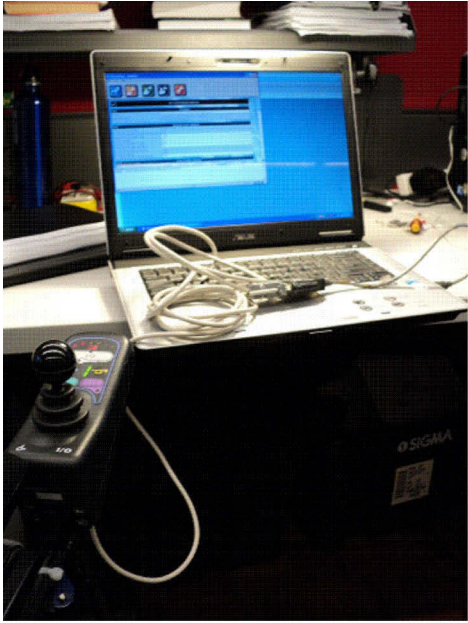


Figure A.6: Hand held programmer setup and parameter samples

Firstly, a programmer is plugged into the DX master as in the Figure A.6, and then a password is needed to access the technician mode of the system. After that, the user can tune parameters of interest at will. The figure A.6 also provides some initial parameters of

the process.

The second method to program DX system is by using software Wizard. The latest version software, Wizard5 can be downloaded from the DX controller website. Figure A.7 illustrates using Wizard5 to program DX master. Software should be installed into the PC prior to use. Additional USB-to RS-232 converter will be needed if PC does not have RS-232 COM port. Furthermore Wizard needs a dongle to program the DX controller. Without a dongle, Wizard can only read the parameters from the controller. The 25-pin dongle is used in our system which will be connected into a parallel port of the PC. Figure A.7 shows the access of the parameters using the Wizard5.



DX UCM Master Remote						
Program						
Name	AM1preslite+1act+CHINcontrol+G80asATT					
Identification	Oz Motors, 40A, 104mOhm, 3 speed					
Master Remote Type (RevA)	Standard UCM Remote					
General User Options						
User Profile Options						
	Prof 1	Prof 2	Prof 3	Prof 4	Prof 5	ACU
Joystick Source (RJM)	RJM	RJM	RJM	Master	Master	
Neutral Maximum (%)	10					
Joystick Swap Forward/Reverse	No	No	No	No	No	No
Sleep Mode Enable	No					
Sleep Timeout (min)	5	5	5	5	5	5
Drive Profiles						
	Prof 1	Prof 2	Prof 3	Prof 4	Prof 5	ACU
Forward Speed @ Maximum (%)	30	60	100	30	55	60
Forward Acceleration (%)	20	30	40	15	20	30
Forward Deceleration (%)	55	70	85	50	60	55
Reverse Speed @ Maximum (%)	25	50	75	20	30	45
Reverse Acceleration (%)	15	25	30	15	20	30
Reverse Deceleration (%)	55	70	80	50	60	55
Turning Speed @ Maximum (%)	15	30	40	15	20	30
Turning Acceleration (%)	10	20	30	10	15	20
Turning Deceleration (%)	60	71	81	50	55	40
Damping Point	45	50	60	40	50	60
Soft-Start Acceleration	No					
Soft-Start Time (ms)	0	0	0	0	0	0
Min to Max Decel Ratio (%)	20					
System Settings						
System Slowdown (%)	20					
Neutral to Parkbrake Delay (ms)	100					
UCM Joystick Swap Left/Right	No					
ACU Enable	Yes					
ACU Joystick Swap Left/Right	Yes					
RJM Enable	Yes					
RJM Joystick Swap Left/Right	Yes					

Figure A.7: Wizard 5 setup and program parameters for DX master

Third method to configure the DX Master is by using HPP plug-in emulator from Wizard5. To start HPP Plug-in, in Wizard5 click Tool -> Plug Ins. System will simulate DX-HPP in the PC as in figure A.8 below.

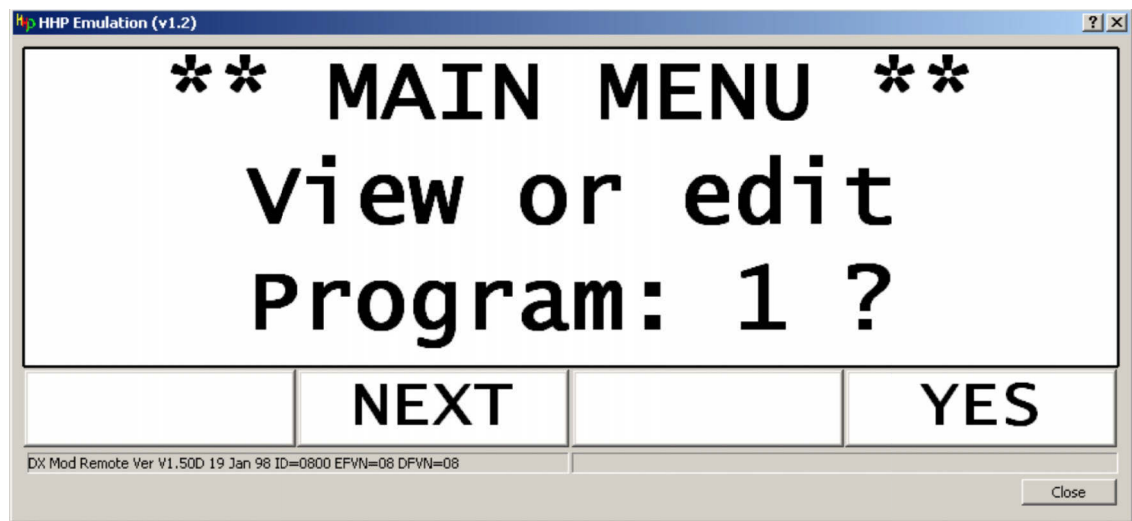


Figure A.8: HPP Plug-in setup

Appendix B

Software Code

B.1 Bayesian neural network training function.

```
function [net,tr] = trainbr_evidence(net,tr,trainV,valV,testV,varargin)

%This neural network training function allows to estimate evidences for
%different networks using Bayesian framework. This work based on a
trainbr %function in neural network %toolbox ver 6.0.3 (R2009b).

%In order to get evidence of the network, use the following structure:
%evidence = tr.evid : This value is only available when the algorithm
%truly converges, that means "Maximum MU reached" message appear. When
%Converge is impossible, the value net.userdata can use to determine the
%approximation of evidence value.

% Modification is performed by this equation:

%          Evid          =          -          perf          -          0.5*log(detA)-
%0.5*numParameters*log(alpha)+0.5*numErrors*log(beta)+log(factorial(h))+2
%*log(h)+0.5*log(2/gamk)+0.5*log(2/(numErrors-gamk));

% where:

%      *) Evid ~ P(D/Hi): This data-dependent term is evidence for model
Hi,

%      *) perf ~ alpha*Ew+beta*ED : Performance function

%      *) A =beta*jj+ii*alph;    Hessian matrix

%      *) numParameters ~ W: number of parameters of the network, including
%both weights and biases.

%      *)h ~ M: number of hidden nodes (or hidden units)of the network.

%      *)numErrors ~ N :number of errors (tn-y(xn,w)) or number of data.

% NOTE: This function uses Levenberg-Marquardt optimization, thus,
%sometimes hessian matrix is singular; consequently tr.evid will get
%infinite value

% Modification data:      20/7/2011
```

```
%% Info

if strcmp(net,'info')

    info.function = mfilename;

    info.title = 'Bayesian Regulation';

    info.type = 'Training';

    info.version = 6;

    info.training_mode = 'Supervised';

    info.gradient_mode = 'Jacobian';

    info.uses_validation = true;

    info.param_defaults.show = 25;

    info.param_defaults.showWindow = true;

    info.param_defaults.showCommandLine = false;

    info.param_defaults.epochs = 1000;

    info.param_defaults.time = inf;

    info.param_defaults.goal = 0;

    info.param_defaults.max_fail = 6;

    info.param_defaults.mem_reduc = 1;

    info.param_defaults.min_grad = 1e-10;

    info.param_defaults.mu = 0.005;

    info.param_defaults.mu_dec = 0.1;

    info.param_defaults.mu_inc = 10;

    info.param_defaults.mu_max = 1e10;

    info.training_states = ...

        [ ...

training_state_info('gradient','Gradient','continuous','log') ...

        training_state_info('mu','Mu','continuous','log') ...

        training_state_info('val_fail','Validation
Checks','discrete','linear') ...

        training_state_info('gamk','Num Parameters','continuous','linear')
...

        training_state_info('ssX','Sum Squared Param','continuous','log') ...
```

```

    ];

    net = info;

    return
end

%% NNET 5.1 Backward Compatibility
if ischar(net)

    switch (net)

        case 'name', info = feval(mfilename,'info'); net = info.title;

        case 'pnames', info = feval(mfilename,'info'); net =
fieldnames(info.param_defaults);

        case 'pdefaults', info = feval(mfilename,'info'); net =
info.param_defaults;

        case 'gdefaults', if (tr==0), net='calcjx'; else net='calcjxbt'; end

        otherwise, error('NNET:Arguments','Unrecognized code.')

    end

    return
end

% CALCULATION
% =====
% Constants

epochs = net.trainParam.epochs;

goal = net.trainParam.goal;

max_fail = net.trainParam.max_fail;

mem_reduc = net.trainParam.mem_reduc;

min_grad = net.trainParam.min_grad;

mu = net.trainParam.mu;

mu_inc = net.trainParam.mu_inc;

mu_dec = net.trainParam.mu_dec;

mu_max = net.trainParam.mu_max;

show = net.trainParam.show;

max_time = net.trainParam.time;

```

```
net.performFcn = 'sse';

% Parameter Checking

if (~isa(epochs,'double')) || (~isreal(epochs)) || (any(size(epochs)) ~=
1) || ...

    (epochs < 1) || (round(epochs) ~= epochs)

    error('NNET:Arguments','Epochs is not a positive integer.')
end

if (~isa(goal,'double')) || (~isreal(goal)) || (any(size(goal)) ~= 1) ||
...

    (goal < 0)

    error('NNET:Arguments','Goal is not zero or a positive real value.')
end

if (~isa(max_fail,'double')) || (~isreal(max_fail)) ||
(any(size(max_fail)) ~= 1) || ...

    (max_fail < 1) || (round(max_fail) ~= max_fail)

    error('NNET:Arguments','Max_fail is not a positive integer.')
end

if (~isa(mem_reduc,'double')) || (~isreal(mem_reduc)) ||
(any(size(mem_reduc)) ~= 1) || ...

    (mem_reduc < 1) || (round(mem_reduc) ~= mem_reduc)

    error('NNET:Arguments','Mem_reduc is not a positive integer.')
end

if (~isa(min_grad,'double')) || (~isreal(min_grad)) ||
(any(size(min_grad)) ~= 1) || ...

    (min_grad < 0)

    error('NNET:Arguments','Min_grad is not zero or a positive real
value.')
end

if (~isa(mu,'double')) || (~isreal(mu)) || (any(size(mu)) ~= 1) || ...

    (mu <= 0)

    error('NNET:Arguments','Mu is not a positive real value.')
end
```

```

if (~isa(mu_dec,'double')) || (~isreal(mu_dec)) || (any(size(mu_dec)) ~=
1) || ...

(mu_dec < 0) || (mu_dec > 1)

error('NNET:Arguments','Mu_dec is not a real value between 0 and 1.')

end

if (~isa(mu_inc,'double')) || (~isreal(mu_inc)) || (any(size(mu_inc)) ~=
1) || ...

(mu_inc < 1)

error('NNET:Arguments','Mu_inc is not a real value greater than 1.')

end

if (~isa(mu_max,'double')) || (~isreal(mu_max)) || (any(size(mu_max)) ~=
1) || ...

(mu_max <= 0)

error('NNET:Arguments','Mu_max is not a positive real value.')

end

if (mu > mu_max)

error('NNET:Arguments','Mu is greater than Mu_max.')

end

if (~isa(show,'double')) || (~isreal(show)) || (any(size(show)) ~= 1) ||
...

(isfinite(show) && ((show < 1) || (round(show) ~= show)))

error('NNET:Arguments','Show is not 'NaN' or a positive integer.')

end

if (~isa(max_time,'double')) || (~isreal(max_time)) ||
(any(size(max_time)) ~= 1) || ...

(max_time < 0)

error('NNET:Arguments','Time is not zero or a positive real value.')

end

% Initialize
Q = trainV.Q;
TS = trainV.TS;
vperf = NaN;
tperf = NaN;

```

```

val_fail = 0;

startTime = clock;

X = getx(net);

numParameters = length(X);

ii = sparse(1:numParameters,1:numParameters,ones(1,numParameters));

% Initialize Performance

original_net = net;

best_net = net;

doValidation = ~isempty(valV.indices);

doTest = ~isempty(testV.indices);

[ssE,El,trainV.Y,Ac,N,Zb,Zi,Zl] =
calcperf2(net,X,trainV.Pd,trainV.Tl,trainV.Ai,Q,TS);

% Initialize regularization parameters

numErrors = 0;

for i=1:size(El,1)
    for j=1:size(El,2)
        numErrors = numErrors + numel(El{i,j});
    end
end

gamk = numParameters;

if ssE == 0,
    beta = 1;
else
    beta = (numErrors - gamk)/(2*ssE);
end

if beta<=0,
    beta=1;
end

ssX = X'*X;

alph = gamk/(2*ssX);

```

```

perf = beta*ssE + alph*ssX;

best_perf = ssE;

if (doValidation)

    [vperf,ignore,valV.Y]
    calcperf2(net,X,valV.Pd,valV.Tl,valV.Ai,valV.Q,valV.TS);

    best_vperf = vperf;

end

%% Training Record

tr.best_epoch = 0;

tr.goal = goal;

tr.states
{'epoch','time','perf','vperf','tperf','mu','gradient','val_fail','gamk',
'ssX'};

%% Status

status = ...

[ ...

training_status('Epoch','iterations','linear','discrete',0,epochs,0), ...
training_status('Time','seconds','linear','discrete',0,max_time,0), ...

training_status('Performance','', 'log','continuous',best_perf,goal,best_p
erf) ... training_status('Gradient','', 'log','continuous',1,min_grad,1)
...

training_status('Mu','', 'log','continuous',mu,mu_max,mu) ...

training_status('Validation
Checks','', 'linear','discrete',0,max_fail,0) ...

training_status('Num Parameters','', 'linear','continuous',gamk,NaN,NaN)
...

training_status('Sum Squared Param','', 'log','continuous',ssX,NaN,NaN)
...

];

nn_train_feedback('start',net,status);

% Train

for epoch=0:epochs

    %You can also tell that the algorithm has converged if the sum squared

    %error (SSE) and sum squared weights (SSW) are relatively constant over
    several iterations

```

```

    if (strcmp(class(net.userdata),'double')&&(net.userdata>2))

        ssE_ssX(epoch+1,1)=ssE;

        ssE_ssX(epoch+1,2)=ssX;

        if (epoch>=100)

            percentage_ssE_ssX=zeros(10,2);

            for i=(epoch-11):(epoch-1)

                percentage_ssE_ssX(i,1)=(ssE_ssX(i+1,1)-
ssE_ssX(i,1))/ssE_ssX(i,1)*100;

                percentage_ssE_ssX(i,2)=(ssE_ssX(i+1,2)-
ssE_ssX(i,2))/ssE_ssX(i,2)*100;

            end

            cout_ssE_ssX=0;

            for i=1:10

                if
((percentage_ssE_ssX(i,1)<0.5)&&(percentage_ssE_ssX(i,2)<0.5))

                    cout_ssE_ssX=cout_ssE_ssX+1;

                else cout_ssE_ssX=0;

                end;

                if (cout_ssE_ssX==10)

                    mu_max=0;

                    ssE_ssX

                end

            end

        end

    end;

    % Jacobian

    [je,jj,gradient]
calcjej(j(net,trainV.Pd,Zb,Zi,Zl,N,Ac,El,Q,TS,mem_reduc);

    % Stopping Criteria

    current_time = etime(clock,startTime);

    [userStop,userCancel] = nntraintool('check');

    if userStop, tr.stop = 'User stop.'; net = best_net;

```

=

```

    elseif userCancel, tr.stop = 'User cancel.'; net = original_net;

    elseif (ssE <= goal), tr.stop = 'Performance goal met.'; net =
best_net;

    elseif (epoch == epochs), tr.stop = 'Maximum epoch reached.';

        net = best_net;

    elseif (current_time >= max_time), tr.stop = 'Maximum time elapsed.';
net = best_net;

    elseif (gradient <= min_grad), tr.stop = 'Minimum gradient reached.';
net = best_net;

    elseif (mu >= mu_max), tr.stop = 'Maximum MU reached.'; net = best_net;

    elseif (doValidation) && (val_fail >= max_fail), tr.stop = 'Validation
stop.'; net = best_net;

end

    % Training record

    if doTest
[tpperf,ignore,testV.Y]=calcperf2(net,X,testV.Pd,testV.Tl,testV.Ai,testV.Q
,testV.TS);

    end

    tr = tr_update(tr,[epoch current_time ssE vperf tperf mu gradient
val_fail gamk ssX]);

    % Feedback

    nn_train_feedback('update',net,status,tr,{trainV valV testV}, ...
[epoch,current_time,best_perf,gradient,mu,val_fail,gamk,ssX]);

    % Stop

    if ~isempty(tr.stop), break, end

    % APPLY LEVENBERG MARQUARDT: INCREASE MU TILL ERRORS DECREASE

    while (mu <= mu_max)

        % CHECK FOR SINGULAR MATRIX

        [msgstr,msgid] = lastwarn;

        lastwarn('MATLAB:nothing','MATLAB:nothing')

        warnstate = warning('off','all');

        dX = -(beta*jj + ii*(mu+alph)) \ (beta*je + alph*X);

        alphI=ii*alph;
%ADD

```

```

[msgstr1,msgid1] = lastwarn;

flag_inv = isequal(msgid1,'MATLAB:nothing');

if flag_inv, lastwarn(msgstr,msgid); end;

warning(warnstate);

X2 = X + dX;

ssX2 = X2'*X2;

net2 = setx(net,X2);

[ssE2,E2,Y2,Ac2,N2,Zb2,Zi2,Zl2] =
calcperf2(net2,X2,trainV.Pd,trainV.Tl,trainV.Ai,Q,TS);

perf2 = beta*ssE2 + alph*ssX2;

if (perf2 < perf) && ( ( sum(isinf(dX)) + sum(isnan(dX)) ) == 0 ) &&
flag_inv
    X = X2; net = net2; Zb = Zb2; Zi = Zi2; Zl = Zl2;

    N = N2; Ac = Ac2; trainV.Y = Y2; El = E2; ssE = ssE2;

    ssX = ssX2;

    mu = mu * mu_dec;

    perf = perf2;

    if (mu < 1e-20), mu = 1e-20; end

    break
end

mu = mu * mu_inc;

end

if (mu <= mu_max)

    warnstate = warning('off','all');

    gamk = numParameters - alph*trace(inv(beta*jj+ii*alph));

    A=beta*jj+ii*alph;

    warning(warnstate);

    if ssX==0,

        alph = 1;

    else

        alph = gamk/(2*(ssX));

```

```

end

if ssE==0,
    beta = 1;
else
    beta = (numErrors - gamk)/(2*ssE);
end

perf = beta*ssE + alph*ssX;

% Validation

if (doValidation)

    [vperf,ignore,valV.Y]
    calcperf2(net,X,valV.Pd,valV.Tl,valV.Ai,valV.Q,valV.TS);

    if (vperf < best_vperf)

        best_net = net;

        best_perf = ssE;

        best_vperf = vperf;

        tr.best_epoch = epoch+1;

        val_fail = 0;

    elseif (vperf > best_vperf)

        val_fail = val_fail + 1;

    end

elseif (ssE < best_perf)

    best_net = net;

    best_perf = ssE;

    tr.best_epoch = epoch+1;

end

end

end

% Evidence estimation

% it is important to let the algorithm run until the effective number of

% parameters has converged. The training might stop with the message

"Maximum MU reached."

```

```
%This is typical, and is a good indication that the algorithm has truly
converged.

% perf1 = beta*ssE + alph*ssX;

% savefile = 'Alphdata.mat';

% save(savefile, 'perf','best_perf','perf1');

if (strcmp(tr.stop, 'Maximum MU reached.'))

    Lamda = eig(A) + diag(alphI);

    detA = 1;

    for i = 1:size(Lamda)

        if (Lamda(i)>0) detA = detA*Lamda(i); end

    end

    if ((detA <= 0) | (detA == -Inf))    detA = 1e-100;    end

    h = net.layers{1}.size;

    evi_alph = 0;

    evi_gamk = 0;

    Evid      =      -      perf      -      0.5*log(detA)      -
0.5*numParameters*log(alph)+0.5*numErrors*log(beta)+...

log(factorial(h))+2*log(h)+0.5*log(2/gamk)+0.5*log(2/(numErrors-gamk));

    tr.evid = Evid;

%-----

end;

% Finish

tr = tr_clip(tr);
```

B. 2 Control Software Code

```
#include <tcpsupp.h>
#include "real-time map.h"
#include <formatio.h>
#include <ansi_c.h>
#include "asynctmr.h"
#include <cvirte.h>
```

```

#include <userint.h>
#include <utility.h>
#include "Real-time map variables.h"
#include "USB4_Functions.h"
#include "Bayesian_Functions.h"
#include <rs232.h>
#define TCPPORT 55555
#define TCPTIMEOUT -1
#define D_MAX 171 // 171 data index
#define T_MAX 349
#define SAFE_BOUND 250 // Safety boundary is around 25 cm
#define RANGE 40 // Effective range is about 40 degrees
#define MAX_DISTANCE 4096 // Maximun range to scan is 4 m
#define ANGLE_START 64 // Start from index 64
#define FILENAME "data.xls"
// TCP IP functions
#define tcpChk(f) if ((g_TCPErrors(f)) < 0) {ReportTCPErrors(); goto Done;} else
int CVICALLBACK ServerTCPCB (unsigned handle, int event, int error,
void *callbackData);

static void ReportTCPErrors (void);
static unsigned int g_hconversation;
static unsigned int isConnected;
static unsigned int isDataReady;
static int g_TCPErrors = 0;
static int aboutpanel;
void CloseProgram(void);
void ControlStateIndicator(int Mode) ;
// Command data from iPhone/iPad
static int Headpanel;
char CommandData[256] = {0};
int HEAD_MOVEMENT_COM_PORT, Port_Status;
//int 3333
static int Headcalpanel;
static int Thres_L = 1200;//1000; // valy > Start_L, receive turn left
command
static int Thres_R = -800;//-1000; // valy < Start_R, receive turn right
command
static int Thres_F = -640;//0; // valy > Start_L, receive turn left
command
static int Thres_B = -2640;//-2190; // valy < Start_R, receive turn
right command
const int Margin_Space = 1000;
// Record data
int newfile;
int USB4_Status;
int Record=0;

int main (int argc, char *argv[])
{
    char tempBuf[256] = {0};
    int registered = 0;

    if (InitCVIRTE (0, argv, 0) == 0)
        return -1; /* out of memory */

```

```

Port_Status =
SearchCOMPORT(&LASER_COM_PORT,&HEAD_MOVEMENT_COM_PORT);
    if (Port_Status == -1){
        MessagePopup ("LASER URG-04LX ERROR ", "Check Laser URG-04LX
connection to computer or it's proper driver");
        return -1;
    }
    // open file to write
    newfile = OpenFile (FILENAME, VAL_WRITE_ONLY, VAL_TRUNCATE,
VAL_ASCII);
    USB4_Status = Init_Config_USB4();
    if (USB4_Status != 0){
        MessagePopup ("ENCODER ERROR", "Check Encoder connection to
computer or it's proper driver");
        return -1;
    }
    // Select the man-machine interface by default, other types can be
selected via software interface.
    Interface_Mode = HEAD_MOVEMENT;
    if ((panelHandle = LoadPanel (0, "real-time map.uir", PANEL)) < 0)
        return -1;
    if ((Headpanel = LoadPanel (0, "real-time map.uir", PANEL_HEAD)) <
0)
        return -1;
    if ((Headcalpanel = LoadPanel (0, "real-time map.uir", PANEL_CAL))
< 0)
        return -1;
    // Load about panel
    if ((aboutpanel = LoadPanel (0, "real-time map.uir", ABOUTPNL)) <
0)
        return -1;
    // For the iphone/ipad interface.
    //TCP IP Initilisation
    DisableBreakOnLibraryErrors();
    // Attempt to register as a TCP server...
    SetWaitCursor (1);
    if (RegisterTCPServer (TCPPOINT, ServerTCPCB, 0) < 0)
        MessagePopup("TCP Server", "Server registration failed!");
    else
    {
        registered = 1;
        SetWaitCursor (0);
        SetCtrlVal (panelHandle, PANEL_ONLINE, 1);
        tcpChk (GetTCPHostAddr (tempBuf, 256));
        SetCtrlVal (panelHandle, PANEL_STATEINDICATOR, "Welcome to SEGWAY
RMP Control Program\n");
        SetCtrlVal (panelHandle, PANEL_STATEINDICATOR, "Developed by
Centre for Health Technologies\n");
        SetCtrlVal (panelHandle, PANEL_STATEINDICATOR, "University
of Technology, Sydney\n");
        SetCtrlVal (panelHandle, PANEL_STATEINDICATOR, "Server
IP:");
        SetCtrlVal (panelHandle, PANEL_STATEINDICATOR, tempBuf);
        SetCtrlVal (panelHandle, PANEL_IPADDRESS,tempBuf);
        tcpChk (GetTCPHostName (tempBuf, 256));
        SetCtrlVal (panelHandle, PANEL_STATEINDICATOR, "\nServer name:");
    }

```

```

        SetCtrlVal (panelHandle, PANEL_STATEINDICATOR, tempBuf);
        SetCtrlVal (panelHandle, PANEL_CONNECTED, 0);
    }
    DisplayPanel (panelHandle);
    Initial_Parameters();
    RunUserInterface ();
    DiscardPanel (panelHandle);
    CloseProgram();
Done:
    if (registered)
        UnregisterTCPServer (TCPPOINT);
    return 0;
}

void Initial_Parameters(void)
{
    int i;
    User_signal *Pointer2;
    // Initialize devices
    //1. Laser range finder URG-04LX
    LaserDataSize = InitURG (LASER_COM_PORT, SCAN_RES, ANGLE_B,
    ANGLE_E, command);
    if (Interface_Mode == HEAD_MOVEMENT)
        OpenHeadMovementPort (HEAD_MOVEMENT_COM_PORT);
    //2. Timers
    NewAsyncTimer (LASER_DAQ_TIMMING, -1, 1, LaserdataDAQ, NULL);
    NewAsyncTimer (MAP_2D_BUILD_TIMMING, -1, 1, Build2DMap, NULL);
    NewAsyncTimer (DISPLAY_FUNCTION_TIMMING, -1, 1, Function_Display, NULL);
    NewAsyncTimer (FREESPACE_FUNCTION_TIMMING, -1, 1,
    Free_space_determination, NULL);
    NewAsyncTimer (READCOMMAND_FUNCTION_TIMMING, -1, 1,
    Read_user_command, NULL);
    SuspendAsyncTimerCallbacks ();
    //3. Encoders
    ReturnModuleAddressUSB1 (0, &ModAddress);
    ResetAllIncPositionsUSB1 (ModAddress);
    //4. Create virtual tasks for USB 6008
    // To read turn right and forward signals
    CreateRead_turn_right_task(&right_task);
    CreateRead_forward_task(&forward_task);
    // To write velocity and steering signals
    CreateWrite_velocity_task(&velocity_task);
    CreateWrite_steering_task(&steering_task);
    // 5. Load neural network parameters
    Function_LoadFFNNNetPara (&netstruct);
    //6. Initialize knock for protected data
    InitializePolarLdata ();
    InitializeEncoder_estimation ();
    InitializeMap2D ();
    InitializeCompensative_map2D_polar ();
    InitializeSignal ();
    InitializeUser_intention ();
    InitializeReady_interface ();
    InitializeGoal ();
    // 7. Choose wheelchair or RMP
    DimentionConfiguration (RECTANGLE);
}

```

```

        for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
            Laser_polar_form[i] =LASER_RANGE_MAX/1000;
        }
// 9. Wheelchair dimentions for display
wheelchairX[0] = -0.1;
wheelchairY[0] = -0.15;
wheelchairX[1] = -0.1;
wheelchairY[1] = 0.15;
wheelchairX[2] = 0.1;
wheelchairY[2] = 0.15;
wheelchairX[3] = 0.1;
wheelchairY[3] = -0.15;
wheelchairX[4] = -0.1;
wheelchairY[4] = -0.15;
// 10. Update control state
ControlStateIndicator(0);
Pointer2 = GetPointerToUser_intention();
    Pointer2->direction = COMMAND_NULL;
    Pointer2->forward_signal = 2.5;
    Pointer2->right_signal = 2.5;
    ReleasePointerToUser_intention();
// 11. Initialize mode machine
    Machine_Mode = 2;
}

void CloseProgram(void)
{
//SetCtrlAttribute (panelHandle, PANEL_TIMER, ATTR_ENABLED, 0);
    DiscardAsyncTimer (-1);
    CloseURG(LASER_COM_PORT);
    if (Interface_Mode == HEAD_MOVEMENT)
        CloseURG (HEAD_MOVEMENT_COM_PORT);
    Exit_USB4();
    CloseFile(newfile);
    MotorControl (velocity_task,steering_task,0,0);
// Clear virtual task
    DAQmxClearTask(velocity_task);
    DAQmxClearTask(steering_task);
    DAQmxClearTask(forward_task);
    DAQmxClearTask(right_task);
// Uninitialize thread safe data
    UninitializePolarLdata();
    UninitializeEncoder_estimation();
    UninitializeMap2D();
    UninitializeCompensative_map2D_polar();
    UninitializeSignal();
    UninitializeUser_intention();
    UninitializeReady_interface();
    UninitializeGoal();
}
int CVICALLBACK Switch_Machine_Mode (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:

```

```

        GetCtrlVal(panelHandle, PANEL_MODE, &Machine_Mode);
        break;
    }
    return 0;
}

int CVICALLBACK Choose_Interface (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            GetCtrlVal(panelHandle,
PANEL_INTERFACE, &Interface_Mode);
            break;
    }
    return 0;
}

int CVICALLBACK Record_Data (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            GetCtrlVal(panelHandle, PANEL_RECORD, &Record);
            break;
    }
    return 0;
}

int CVICALLBACK Head_Test (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            DisplayPanel(Headpanel);
            SetCtrlAttribute (Headpanel, PANEL_HEAD_TIMER_HEAD,
ATTR_ENABLED, 1);
            break;
    }
    return 0;
}

int CVICALLBACK Quit_Head (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            SetCtrlAttribute (Headpanel, PANEL_HEAD_TIMER_HEAD, ATTR_ENABLED, 0);
            HidePanel(Headpanel);
            break;
    }
    return 0;
}

```

```

}

int CVICALLBACK Quit_Cali (int panel, int control, int event,
                          void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            SetCtrlAttribute (Headpanel,
            PANEL_CAL_TIMER_CALIBRATION, ATTR_ENABLED, 0);
            HidePanel (Headcalpanel);
            break;
    }
    return 0;
}

int CVICALLBACK Head_Calibrate (int panel, int control, int event,
                              void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            DisplayPanel (Headcalpanel);
            break;
    }
    return 0;
}

int CVICALLBACK Read_head_movement_data (int panel, int control, int
event,
                                         void *callbackData, int eventData1, int eventData2)
{
    double Forward_signal, Right_signal;
    double Forward_signal_F, Right_signal_F;
    int F_B_raw, R_L_raw;
    int toggleButton = 0;
    static int RL_tem=0, FB_tem=0, Count =0, Mean_RL=0, Mean_FB =0;
    int RLraw_F, FBraw_F;
    int RL_dif, FB_dif;
    char buffer[50];
    switch (event)
    {
        case EVENT_TIMER_TICK:
            GetRawHeadMovementData (HEAD_MOVEMENT_COM_PORT, &R_L_raw, &F_B_raw);
            RawHeadDataConvert ( R_L_raw, F_B_raw, Thres_L, Thres_R, Thres_F, Thres_B,
            &Forward_signal, &Right_signal);
            Head_Movement_Filter (Forward_signal,
            Right_signal, &Forward_signal, &Right_signal);
            Head_Movement_Noise_Filter (Forward_signal,
            Right_signal,
            &Forward_signal_F, &Right_signal_F);
            SetCtrlVal (Headpanel,
            PANEL_HEAD_NUMERIC_4, Right_signal_F);
            SetCtrlVal (Headpanel, PANEL_HEAD_NUMERIC_4, Right_signal_F);
            SetCtrlVal (Headpanel, PANEL_HEAD_NUMERIC_5, Forward_signal_F);
            PlotStripChartPoint (Headpanel,
            PANEL_HEAD_STRIPCHART_R, Right_signal);
    }
}

```

```

PlotStripChartPoint      (Headpanel,          PANEL_HEAD_STRIPCHART_RD,
Right_signal_F);
PlotStripChartPoint (Headpanel, PANEL_HEAD_STRIPCHART_F, Forward_signal);
PlotStripChartPoint      (Headpanel,
PANEL_HEAD_STRIPCHART_FD,Forward_signal_F);
GetCtrlVal (Headcalpanel,
PANEL_CAL_TOGGLEBUTTON,&toggleButton);
if (toggleButton){
    RL_dif = R_L_raw-RL_tem;
    FB_dif = F_B_raw-FB_tem;
    if ((abs(RL_dif)<500)&&(abs(FB_dif)<500)){// ensure signals stable
        Mean_RL = Mean_RL+R_L_raw;
        Mean_FB = Mean_FB+ F_B_raw;
        Count++;
    };
    RL_tem = R_L_raw;
    FB_tem = F_B_raw;
    if (Count>3){
Thres_L = RoundRealToNearestInteger ((double)Mean_RL/Count+Margin_Space);
Thres_R = RoundRealToNearestInteger ((double)Mean_RL/Count-Margin_Space);
Thres_F = RoundRealToNearestInteger ((double)Mean_FB/Count+Margin_Space);
Thres_B = RoundRealToNearestInteger ((double)Mean_FB/Count-Margin_Space);
SetCtrlVal (Headcalpanel, PANEL_CAL_TOGGLEBUTTON,0);
        Count =0;
    }

    } else {
        Count =0;
        Mean_RL=0;
        Mean_FB=0;
        RL_tem = 0;
        FB_tem = 0;
    }
    SetCtrlVal (Headcalpanel, PANEL_CAL_NUMERIC_F,Thres_F);
    SetCtrlVal (Headcalpanel, PANEL_CAL_NUMERIC_B,Thres_B);
    SetCtrlVal (Headcalpanel, PANEL_CAL_NUMERIC_R,Thres_R);
    SetCtrlVal (Headcalpanel, PANEL_CAL_NUMERIC_L,Thres_L);
    break;
}
return 0;
}

int CVICALLBACK QuitCallback (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            CloseProgram();
            QuitUserInterface (0);
            break;
        case EVENT_RIGHT_CLICK:
            break;
    }
    return 0;
}

```

```

int CVICALLBACK LaserdataDAQ (int reserved, int theTimerId, int event,
                             void *callbackData, int eventData1,
                             int eventData2)
{
    double laserdataX[MAP_2D_1DEC_SIZE], laserdataY[MAP_2D_1DEC_SIZE];
    int Laserstatus;
    int i;
    Laser_map *Pointer;
    int Ang_B, Ang_E;

    Ang_B = 0;
    Ang_E = 0;
    Laserstatus = LaserDataDAQ (LASER_COM_PORT, LaserDataSize,
    laserdataX, laserdataY, SCAN_RES, command);
    if (Laserstatus){
        // Move laser data from the laser origin into the reference's
origin
        MoveDataXYO ((double)laserX, (double)laserY, 0, laserdataX,
laserdataY,
                    laserdataX, laserdataY, LaserDataSize);
        // Rearrange the data from starting point to ending points in
the polar form
        ReformData1 (laserdataX, laserdataY, Laser_polar_form,
&Ang_B, &Ang_E, LaserDataSize);
    }
    Pointer = GetPointerToPolarLdata();
    Pointer -> Angle_B = Ang_B;
    Pointer -> Angle_E = Ang_E;
    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        Pointer -> PolarLMap[i] = Laser_polar_form[i];
    }
    ReleasePointerToPolarLdata();
    return 0;
}

int CVICALLBACK Build2DMap (int reserved, int theTimerId, int event,
                           void *callbackData, int eventData1,
                           int eventData2)
{
    int i;
    static long PrevPosP1 = 0, PrevPosP2 = 0;
    double DeltaDis, DeltaPhi, temp;
    static double Laser_polar_form1[MAP_2D_1DEC_SIZE];
    Encoder *Pointer;
    Normal_map *Pointer1;
    Laser_map *Pointer2;
    double *Pointer3;
    int Right_Wheel_Increments, Left_Wheel_Increments;
    double Factor;
    double DeltaDis, DeltaPhi;
    char buffer[50];
    double Velocity;
    double dX, dY, dO;
    int Ang_B, Ang_E;
    static int k = 0;

```

```

        static double Laser2DMapX[MAP_2D_1DEC_SIZE];    //Real odometry map
before transform
        static double Laser2DMapY[MAP_2D_1DEC_SIZE];
        static double Laser2DMapX1[MAP_2D_1DEC_SIZE];    //Real odometry map
before transform
        static double Laser2DMapY1[MAP_2D_1DEC_SIZE];

// 1.initial value for laser variables
if (k==0){
    k=k+1; // Only initialize the map data once for whole running
time.

    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        Laser2DMapX[i] = 4.096*cos(i*2*PI/360);
        Laser2DMapY[i] = 4.096*sin(i*2*PI/360);
    }
}
// 2. Get lock to access PolarLdata
Pointer2 = GetPointerToPolarLdata();
Ang_B = Pointer2 -> Angle_B;
Ang_E = Pointer2 -> Angle_E;
    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        Laser_polar_form1[i]= Pointer2 -> PolarLMap[i];
    }
ReleasePointerToPolarLdata();
// 3. Require the encoder data.
Get_Wheel_Increments_USB4(&Right_Wheel_Increments,
&Left_Wheel_Increments);
Factor = rearwheelDim*3.1415926535/3600;
DeltaDis = (Right_Wheel_Increments +
Left_Wheel_Increments)*Factor/1000/2; // unit (m)
DeltaPhi = (Right_Wheel_Increments -
Left_Wheel_Increments)*Factor/rearaxislength;
// 4. Calculate velocity and steering angle of wheelchair
dX =-DeltaDis*sin(DeltaPhi/2)*1000;
dY = DeltaDis*cos(DeltaPhi/2)*1000;
dO = DeltaPhi;
// 5. Update 2D map in when new positon based on the encoders'
measure
MoveDataXYO (0, 0, dO, Laser2DMapX, Laser2DMapY, Laser2DMapX,
Laser2DMapY, MAP_2D_1DEC_SIZE);
MoveDataXYO (dX, dY, 0, Laser2DMapX, Laser2DMapY,
Laser2DMapX, Laser2DMapY, MAP_2D_1DEC_SIZE);
ReformData (Laser2DMapX, Laser2DMapY, polar2Dmap,
MAP_2D_1DEC_SIZE);
// 6. Build real -time 2D map
Buildnew2Dmap (Laser_polar_form1, polar2Dmap,Laser2DMapX,
Laser2DMapY,Ang_B,Ang_E);
    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        Laser2DMapX[i] = Laser2DMapX[i];
        Laser2DMapY[i] = Laser2DMapY[i];
    }
// 7. Compensate the wheelchair dimension
if (compensatingmode == RECTANGLE)
    RecDimComp2Dmap (Laser2DMapX, Laser2DMapY, polar2Dmap,
leftbound, rightbound,frontbound, rearbound, safewidth, safelength);
if (compensatingmode == CIRCLE)

```

```

        CirDimComp2DMap (Laser2DMapX, Laser2DMapY, polar2Dmap,
centerX, centerY, radius, safearea);
    // 8. Record the data into excel file for the experiment
    if (Record){
        Velocity = DeltaDis/0.11; // speed (m/s)
        // write on file
        Fmt (buffer, "%s<%f", Velocity);
        strcat(buffer, " ");
        WriteFile (newfile, buffer, StringLength(buffer));
        DeltaPhi = DeltaPhi/0.11; // turning speed (rad/s)
        // write on file
        Fmt (buffer, "%s<%f", DeltaPhi);
        strcat(buffer, "\n");
        WriteFile (newfile, buffer, StringLength(buffer));
    }
// 6. Copy data to display
    Pointer = GetPointerToEncoder_estimation();
    Pointer->Velocity_encoder=velocity_encod;
    Pointer->Steering_encoder=steering_encod;
    ReleasePointerToEncoder_estimation();
    Pointer1 = GetPointerToMap2D();
    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        Pointer1 -> Normal_Map2DX[i] = Laser2DMapX[i];
        Pointer1 -> Normal_Map2DY[i] = Laser2DMapY[i];
    }
    ReleasePointerToMap2D();
    Pointer3 = GetPointerToCompensative_map2D_polar();
    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        *(Pointer3+i) = polar2Dmap[i];
    }
    ReleasePointerToCompensative_map2D_polar();
    return 0;
}

int CVICALLBACK Function_Display (int reserved, int theTimerId, int
event, void *callbackData, int eventData1, int eventData2)
{
    int i, target;
    double x_mid,y_mid;
    double Nor_Map2DX[MAP_2D_1DEC_SIZE]; //Real odometry map
    double Nor_Map2DY[MAP_2D_1DEC_SIZE];
    double Com_Map2DX[MAP_2D_1DEC_SIZE];
    double Com_Map2DY[MAP_2D_1DEC_SIZE];
    double velocity_encod,steering_encod;
    double Control[2];
    Encoder *Pointer;
    Normal_map *Pointer1;
    Control_signal *Pointer2;
    int *Pointer3;
    double *Pointer4;

    Pointer = GetPointerToEncoder_estimation();
        velocity_encod=Pointer->Velocity_encoder;
        steering_encod=Pointer->Steering_encoder;
    ReleasePointerToEncoder_estimation();
    Pointer1 = GetPointerToMap2D();

```

```

        for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
            Nor_Map2DX[i]=Pointer1 -> Normal_Map2DX[i];
            Nor_Map2DY[i]=Pointer1 -> Normal_Map2DY[i];
        }
    ReleasePointerToMap2D();
    Pointer2 = GetPointerToSignal();
    Control[0]= Pointer2 -> Velocity;
    Control[1]= Pointer2 -> Steering;
    ReleasePointerToSignal();
    Pointer3 = GetPointerToGoal();
    target=*Pointer3;
    ReleasePointerToGoal();
    Pointer4 = GetPointerToCompensative_map2D_polar();
    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        Com_Map2DX[i]=*(Pointer4+i)*cos(i*2*PI/360);
        Com_Map2DY[i]=*(Pointer4+i)*sin(i*2*PI/360);
    }
    ReleasePointerToCompensative_map2D_polar();
    //1. Update 2D Map
    PlotXY (panelHandle, PANEL_GRAPH_2D, Nor_Map2DX, Nor_Map2DY,
    MAP_2D_1DEC_SIZE, VAL_DOUBLE, VAL_DOUBLE, VAL_THIN_LINE, VAL_NO_POINT,
    VAL_SOLID, 1, VAL_BLACK);
    //2. Update 2D Dim Comp Map
    PlotXY (panelHandle, PANEL_GRAPH_2D, Com_Map2DX,
    Com_Map2DY,MAP_2D_1DEC_SIZE, VAL_DOUBLE, VAL_DOUBLE, VAL_THIN_LINE,
    VAL_NO_POINT, VAL_SOLID, 1,
    VAL_BLUE);
    //3. Draw the wheelchair
    PlotXY (panelHandle, PANEL_GRAPH_2D, wheelchairX, wheelchairY, 5,
    VAL_DOUBLE,VAL_DOUBLE, VAL_FAT_LINE, VAL_NO_POINT, VAL_SOLID, 1,
    VAL_MAGENTA);
    PlotLine (panelHandle, PANEL_GRAPH_2D, 0, 0, Com_Map2DX[target],
    Com_Map2DY[target], VAL_RED);
    DeleteGraphPlot (panelHandle, PANEL_GRAPH_2D, -1,
    VAL_DELAYED_DRAW);
    // Display on strip chart
    PlotStripChart (panelHandle, PANEL_SIGNAL_CONTROL_CHART, Control,
    2, 0, 0, VAL_DOUBLE);
    PlotStripChart (panelHandle, PANEL_SPEED_CHART, &velocity_encod, 1,
    0, 0, VAL_DOUBLE);
    return 0;
}

int CVICALLBACK Free_space_determination (int reserved, int theTimerId,
int event,
void *callbackData, int eventData1,int eventData2)
{
    static int target =90;
    static int k=0;
    static int command_state = COMMAND_PERFORMING, preFS_B = 90,
preFS_E = 90;
    double densFW,densR,densL, maxval;
    double threshold;
    double netinput[IN_NUM], netoutput[OUT_NUM];
    double vel,ste;

```

```

    int numFS,curFSIndex;
    double Com_LaserPolar[MAP_2D_1DEC_SIZE];
    int user_command,i;
    double *Pointer;
    int *Pointer3;
    Control_signal *Pointer1;
    User_signal *Pointer2;
    float64 forward_signal, right_signal;
    // for shared control criteria
    int Current_Direction;
    static int Operating_Mode = MACHINE;
    static int Continuous_Command = 0;
    double scale;

    Pointer = GetPointerToCompensative_map2D_polar();
    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        Com_LaserPolar[i]=*(Pointer+i);
    }
    ReleasePointerToCompensative_map2D_polar();
    scale = 1;
    Pointer2 = GetPointerToUser_intention();
    user_command = Pointer2->direction;
    forward_signal = scale* Pointer2->forward_signal;
    right_signal = scale* Pointer2->right_signal;
    ReleasePointerToUser_intention();
    if (Machine_Mode == 3){
        if (user_command!=COMMAND_NULL){
            k=1;
        }
        if(k==1){
            // when stop command occurs
            if (user_command==GO_BACKWARD){
                k=0;
                ste=2.5;
                vel=2.5;
                ManualControlJoyS (velocity_task,steering_task,vel, ste);
            }
        }
        if (k==1){
            ObstacleDensity(Com_LaserPolar,
            MAP_2D_1DEC_SIZE/2,&densFW,&densR,&densL,&maxval);
            threshold=0.85*maxval;
            numFS = Bayesian_Recursive_Freespace(Com_LaserPolar,
            FSspace,threshold);

            curFSIndex=UpdateTargetAuto(preFS_B,preFS_E,numFS,K_auto,&target,FS
            space);
            if ((user_command!=
            COMMAND_NULL)&&(command_state!=COMMAND_DONE))
                command_state =
            Bayesian_Shared_Control_Strategy(FSspace,
            numFS,user_command,K_command,&curFSIndex,&target);
            else
                if (user_command = COMMAND_NULL) command_state =
            COMMAND_PERFORMING;
                preFS_B=FSspace[curFSIndex].FS_B;
                preFS_E=FSspace[curFSIndex].FS_E;

```

```

Function_GetNNetInput      (netinput,      Com_LaserPolar,
densFW,target);
Function_CompOutFFNNet(netinput, netoutput, &netstruct);
// The system starts working.
// Shared control criteria
// Make sure the continuous command is consistent
if (Machine_Mode == SHARE_MODE){
if (Interface_Mode ==JOYSTICK)
{
    if ((Operating_Mode ==
MACHINE)&&((forward_signal>3)||(forward_signal<3)
        ||(right_signal>3)||(right_signal<2)))
        Continuous_Command++;
    //find the direction of travel
    if
((Continuous_Command>=2)&&((forward_signal>2.9)||(forward_signal<2.1)
        ||(right_signal>2.9)||(right_signal<2.1))) {
        Continuous_Command = 0;
        Current_Direction =
RoundRealToNearestInteger(atan2(forward_signal-2.5,      right_signal-
2.5)/(PI/180));
        Environment_Operating_Mode
        (Com_LaserPolar,Current_Direction,&Operating_Mode);
        Operating_Mode = Operating_Mode;
    }
    else
        Operating_Mode = MACHINE;
}
if (Interface_Mode ==HEAD_MOVEMENT)
{
    if ((Operating_Mode ==
MACHINE)&&((forward_signal>3)||(forward_signal<2)||(right_signal>3)||(rig
ht_signal<2)))
        Continuous_Command++;
    //find the direction of travel
    if ((Continuous_Command>=2)&&((forward_signal>2.8)||
        (forward_signal<2.2)||(right_signal>2.8)||(right_signal<2.2))) {
        Current_Direction =
RoundRealToNearestInteger(atan2(forward_signal-2.5,      right_signal-
2.5)/(PI/180));
        Environment_Operating_Mode
        (Com_LaserPolar,Current_Direction,&Operating_Mode);
    }
    else
        Operating_Mode = MACHINE;
}
if ((Operating_Mode == MACHINE)&&(Continuous_Command>=2))
    Continuous_Command = 0;
}
if (Machine_Mode ==AUTONOMOUS_MODE)
    Operating_Mode = MACHINE;
if (Machine_Mode ==MANUAL_MODE)
    Operating_Mode = MANUAL;
if (Operating_Mode==URGENT){
    k=0;

```

```

        Operating_Mode = MACHINE;
        Continuous_Command = 0;
        ste=2.5;
        vel=2.5;
        ManualControlJoyS      (velocity_task,steering_task,vel,
ste);
    }
    if (Operating_Mode == MACHINE){
        vel=netoutput[0];
        ste=netoutput[1]*180/PI;
        if (ste>=90) ste = 89;
        if (ste<=-90) ste = -89;
        if (vel>MAX_SPEED) vel = MAX_SPEED;
        if (vel<-0.3) vel= -0.3;
        ste=-ste;
        vel = vel + 2.53;
        if (vel < 2.5) vel = 2.5;
        if (vel > 3.2) vel = 3.2;
        ste = ste*PI/180;
        if (ste > 0){
            ste = 1/1.9*ste + 2.50;
            if (ste < 2.5) ste = 2.5;
            if (ste > 3.5) ste = 3.5;
        }
        else{
            ste = 1/1.8*ste + 2.45;
            if (ste < 1.5) ste = 1.5;
            if (ste> 2.5) ste = 2.5;
        }
        if (ste == 0) ste = 2.5;
        ManualControlJoyS (velocity_task,steering_task,vel, 5-
ste);

        SetCtrlVal (panelHandle, PANEL_Machine, 1);
        SetCtrlVal (panelHandle, PANEL_Manual, 0);
    }
    if (Operating_Mode == MANUAL){
        // limit translated and steering speed for safety
        if (right_signal>3.2)
            right_signal = 3.2;
        if (right_signal <1.8)
            right_signal = 1.8;
        if (forward_signal>3.2)
            forward_signal = 3.2;
        ManualControlJoyS
(velocity_task,steering_task,forward_signal, right_signal);
        SetCtrlVal (panelHandle, PANEL_Machine, 0);
        SetCtrlVal (panelHandle, PANEL_Manual, 1);
    }
}
}
Pointer1 = GetPointerToSignal();
Pointer1 -> Velocity = 2.5;
Pointer1 -> Steering = 2.5;
ReleasePointerToSignal();
}
if (Machine_Mode == URGENT) {

```

```

    k=0;
    ste=2.5;
    vel=2.5;
    ManualControlJoyS (velocity_task,steering_task,vel, ste);

}
if (Machine_Mode == MANUAL_MODE) {
// limit translated and steering speed for safety
    if (right_signal>3.2)
        right_signal = 3.2;
    if (right_signal <1.8)
        right_signal = 1.8;
    if (forward_signal>3.2)
        forward_signal = 3.2;
    if (forward_signal<1.8)
        forward_signal = 1.8;
    ManualControlJoyS
(velocity_task,steering_task,forward_signal, right_signal);
    SetCtrlVal (panelHandle, PANEL_Machine, 0);
    SetCtrlVal (panelHandle, PANEL_Manual, 1);
}

    Pointer3 = GetPointerToGoal();
    *Pointer3=target;
    ReleasePointerToGoal();
    return 0;
}

int CVICALLBACK Read_user_command (int reserved, int theTimerId, int
event,
    void *callbackData, int eventData1,int eventData2)
{
    float64 forward_signal, right_signal;
    float64 Vforward, Vright;
    int user_command;
    static int Emergency=0, Mode_Change=0, k=0;//Head_status=1;
    User_signal *Pointer;
    int r_l_raw, f_b_raw;

    //1. Read signals from Joystick.
    if (Interface_Mode == JOYSTICK)

        GetJoystickSignal(forward_task,right_task,&forward_signal,&right_si
gnal);

        if (forward_signal<2){
            Emergency = 1;
            k=1;
            if (Mode_Change ==1){
                Machine_Mode = Machine_Mode+1;
                if (Machine_Mode>3)
                    Machine_Mode =1;
                SetCtrlVal(panelHandle, PANEL_MODE,Machine_Mode);
                Mode_Change =0;
            }
        }
    else{

```

```

        Emergency =0;
        if (k==1) Mode_Change =1;
        k=0;
    }
    if (Interface_Mode == HEAD_MOVEMENT){
        GetRawHeadMovementData (HEAD_MOVEMENT_COM_PORT,
        &r_l_raw, &f_b_raw);
        RawHeadDataConvert ( r_l_raw, f_b_raw,Thres_L,Thres_R,
        Thres_F,
        Thres_B, &forward_signal, &right_signal);
        Head_Movement_Filter(forward_signal,
        right_signal,&forward_signal,&right_signal);
        Head_Movement_Noise_Filter(forward_signal,
        right_signal, &forward_signal,&right_signal);
    }
    if ((forward_signal>3)&&(forward_signal<4)){
        user_command = GO_FORWARD;
        ControlStateIndicator(GO_FORWARD); // Stop
    }
    else if((forward_signal<2)&&(forward_signal>1)){
        user_command = GO_BACKWARD;
        ControlStateIndicator(GO_BACKWARD);
    }
    else if ((right_signal>3)&&(right_signal<4)){
        user_command = GO_RIGHT;
        ControlStateIndicator(GO_RIGHT);
    }
    else if ((right_signal<2)&&(right_signal>1)){
        user_command = GO_LEFT;
        ControlStateIndicator(GO_LEFT);
    }
    else {
        user_command = COMMAND_NULL;
        Mode_Change =1;
    }
    SetCtrlVal (panelHandle, PANEL_NUMERIC,right_signal);
    //Vright);
    SetCtrlVal (panelHandle, PANEL_NUMERIC_2,
    forward_signal); //Vforward);
    // Process Changing modes using backward command
    if (Emergency) {
        user_command = GO_BACKWARD;
    }
    //2. Pass user's command to User_intention variable.
    Pointer = GetPointerToUser_intention();
    Pointer->direction = user_command;
    Pointer->forward_signal = forward_signal;
    Pointer->right_signal = right_signal;
    ReleasePointerToUser_intention();
    return 0;
}

int CVICALLBACK Run_Function (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    int *Pointer;
    static int Run_status=0;

```

```

switch (event)
{
    case EVENT_COMMIT:

        if (Run_status == 0){
            Run_status = 1;
            SetCtrlAttribute (panelHandle, PANEL_BUT_RUN,
ATTR_LABEL_TEXT, "STOP");
            ResumeAsyncTimerCallbacks ();
        }
        else{
            Run_status = 0;

            Pointer=GetPointerToReady_interface ();
            *Pointer=0;
            ReleasePointerToReady_interface ();
            MotorControl (velocity_task,steering_task,0,0);
            SetCtrlAttribute (panelHandle, PANEL_BUT_RUN,
ATTR_LABEL_TEXT, "RUN");
            SuspendAsyncTimerCallbacks ();
        }
        break;
    case EVENT_RIGHT_CLICK:
        break;
}
return 0;
}
// User define function
void DimentionConfiguration(int mode)
{
    leftbound=275;
    rightbound=275;
    frontbound=350;
    rearbound=200;
    safewidth=0;
    safelength=0;
    laserX=275;
    laserY=-550;
    centerX=0;
    centerY=300;
    radius=300;
    safearea=0;
    rearwheeldim=350;
    rearaxislength=555;
    switch (mode){
        case CIRCLE: // RMP
            compensatingmode=RECTANGLE;
            break;
        case RECTANGLE: // Wheelchair
            compensatingmode=RECTANGLE;
            break;
    }
}

int CVICALLBACK ServerTCPCB (unsigned handle, int event, int error,

```

```

void *callbackData)
{
    char receiveBuf[256] = {0};
    int dataSize         = sizeof (receiveBuf) - 1;
    char addrBuf[31];
    int user_command;
    User_signal *Pointer;

    switch (event)
    {
        case TCP_CONNECT:
            if (g_hconversation)
            {
                // We already have one client, don't accept another...
                tcpChk (GetTCPPeerAddr (handle, addrBuf, 31));
                sprintf (receiveBuf, "\n-- Refusing conection request
from "
                                "%s --\n", addrBuf);
                SetCtrlVal (panelHandle,
PANEL_STATEINDICATOR, receiveBuf);
                tcpChk (DisconnectTCPClient (handle));
            }
            else
            {
                isConnected=1;
                // Handle this new client connection
                // Enable Update Timer
                SetCtrlAttribute (panelHandle, PANEL_TIMER,
ATTR_ENABLED, 1);
                g_hconversation = handle;
                SetCtrlVal (panelHandle, PANEL_CONNECTED, 1);
                tcpChk (GetTCPPeerAddr (g_hconversation, addrBuf, 31));
                SetCtrlVal (panelHandle,
PANEL_STATEINDICATOR, "\nConnecting to IP:");
                SetCtrlVal (panelHandle,
PANEL_STATEINDICATOR, addrBuf);
                tcpChk (SetTCPDisconnectMode (g_hconversation,
TCP_DISCONNECT_AUTO));
            }
            break;
        case TCP_DATAREADY:
            if ((dataSize = ServerTCPRead (g_hconversation,
receiveBuf, dataSize, 1000)) < 0)
            {
                isDataReady=0;
                SetCtrlVal (panelHandle, PANEL_STATEINDICATOR, "\nTCP IP
Receiving errors:");
            }
            else // Process receiving data in here
            {
                isDataReady=1;
                receiveBuf[dataSize] = '\0';

                if (strcmp (receiveBuf, "STOP", 4) == 0) {
                    ControlStateIndicator (GO_STOP);
                    user_command= GO_BACKWARD;
                }
            }
        }
    }
}

```

```

        }
        else if(strncmp
(receiveBuf,"FASTFORWARD",11)==0){

        ControlStateIndicator(GO_FORWARD);

        user_command= GO_FORWARD;
        }
        else if(strncmp
(receiveBuf,"SLOWFORWARD",11)==0){

        ControlStateIndicator(GO_FORWARD);
        user_command= GO_FORWARD;
        }
        else if(strncmp
(receiveBuf,"TURNRIGHT",9)==0){

        ControlStateIndicator(GO_RIGHT);

        user_command= GO_RIGHT;
        }
        else if(strncmp
(receiveBuf,"TURNLEFT",8)==0){

        ControlStateIndicator(GO_LEFT);
        user_command= GO_LEFT;
        }
        else if(strncmp
(receiveBuf,"BACKWARD",8)==0){

        ControlStateIndicator(GO_BACKWARD);

        user_command= GO_BACKWARD;
        }
        else{
        ControlStateIndicator(-1);
        user_command= COMMAND_NULL;
        }
        if (Interface_Mode == IPHONE){
        Pointer =
GetPointerToUser_intention();

        Pointer->direction = user_command;
        Pointer->forward_signal = 0;
        Pointer->right_signal = 0;
        ReleasePointerToUser_intention();
        }
    } // End of process receiving data

    break;
case TCP_DISCONNECT:
    if (handle == g_hconversation)
    {
        // The client we were talking to has
        disconnected...

        // Stop update timer
        SetCtrlAttribute (panelHandle, PANEL_TIMER,
ATTR_ENABLED, 0);

        SetCtrlVal (panelHandle, PANEL_CONNECTED, 0);
        g_hconversation = 0;
        isConnected=0;
        isDataReady=0;
    }

```

```

        ControlStateIndicator(-1);
        SetCtrlVal                                (panelHandle,
PANEL_STATEINDICATOR, "\nClient was disconnected");
    }
    break;
}

Done:
    return 0;
}
/*****
*****
// Functions' defintion
/*-----*/
/*-----*/
/*          Report          TCP          Errors          if          any
*/
/*-----*/
/*-----*/
static void ReportTCPError (void)
{
    if (g_TCPErrors < 0)
    {
        char    messageBuffer[1024];
        sprintf(messageBuffer,
            "TCP library error message: %s\nSystem error message: %s",
            GetTCPErrorString (g_TCPErrors), GetTCPSystemErrorString());
        MessagePopup ("Error", messageBuffer);
    }
}

void    ControlStateIndicator(int Mode)
{
    switch (Mode){
        case -1:// Disconnection from client
            SetCtrlAttribute    (panelHandle,    PANEL_PICSTOP,
ATTR_DIMMED, 1);
            SetCtrlAttribute    (panelHandle,    PANEL_PICSLOWFORWARD,
ATTR_DIMMED, 1);
            SetCtrlAttribute    (panelHandle,    PANEL_PICTURNRIGHT,
ATTR_DIMMED, 1);
            SetCtrlAttribute    (panelHandle,    PANEL_PICTURNLEFT,
ATTR_DIMMED, 1);
            SetCtrlAttribute    (panelHandle,    PANEL_PICBACKWARD,
ATTR_DIMMED, 1);
            break;
        case 0:// 0: Initialisation
            SetCtrlAttribute    (panelHandle,    PANEL_PICSTOP,
ATTR_DIMMED, 1);
            SetCtrlAttribute    (panelHandle,    PANEL_PICSLOWFORWARD,
ATTR_DIMMED, 1);
            SetCtrlAttribute    (panelHandle,    PANEL_PICTURNRIGHT,
ATTR_DIMMED, 1);
            SetCtrlAttribute    (panelHandle,    PANEL_PICTURNLEFT,
ATTR_DIMMED, 1);

```

```

        SetCtrlAttribute      (panelHandle,      PANEL_PICBACKWARD,
ATTR_DIMMED, 1);
        // Stop update timer
        break;
    case GO_STOP:// 1: Stop
        SetCtrlAttribute      (panelHandle,      PANEL_PICSTOP,
ATTR_DIMMED, 0);
        SetCtrlAttribute      (panelHandle,      PANEL_PICSLOWFORWARD,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICTURNRIGHT,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICTURNLEFT,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICBACKWARD,
ATTR_DIMMED, 1);
        SetCtrlVal (panelHandle, PANEL_STATEINDICATOR, "\nStop
Command Received");
        break;
    case GO_FORWARD:// 2: Slow Forward
        SetCtrlAttribute      (panelHandle,      PANEL_PICSTOP,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICSLOWFORWARD,
ATTR_DIMMED, 0);
        SetCtrlAttribute      (panelHandle,      PANEL_PICTURNRIGHT,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICTURNLEFT,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICBACKWARD,
ATTR_DIMMED, 1);
        SetCtrlVal (panelHandle,      PANEL_STATEINDICATOR,
"\nForward Command Received");
        break;
    case GO_RIGHT:// 3: Turn Right
        SetCtrlAttribute      (panelHandle,      PANEL_PICSTOP,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICSLOWFORWARD,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICTURNRIGHT,
ATTR_DIMMED, 0);
        SetCtrlAttribute      (panelHandle,      PANEL_PICTURNLEFT,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICBACKWARD,
ATTR_DIMMED, 1);
        SetCtrlVal (panelHandle,      PANEL_STATEINDICATOR, "\nTurn
Right Command Received");
        break;
    case GO_LEFT:// 4: Turn left
        SetCtrlAttribute      (panelHandle,      PANEL_PICSTOP,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICSLOWFORWARD,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICTURNRIGHT,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICTURNLEFT,
ATTR_DIMMED, 0);

```

```

        SetCtrlAttribute      (panelHandle,      PANEL_PICBACKWARD,
ATTR_DIMMED, 1);
        SetCtrlVal (panelHandle, PANEL_STATEINDICATOR, "\nTurn
Left Command Received");
        break;
    case GO_BACKWARD:// 6: Backward
        SetCtrlAttribute      (panelHandle,      PANEL_PICSTOP,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICSLOWFORWARD,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICTURNRIGHT,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICTURNLEFT,
ATTR_DIMMED, 1);
        SetCtrlAttribute      (panelHandle,      PANEL_PICBACKWARD,
ATTR_DIMMED, 0);
        SetCtrlVal (panelHandle,      PANEL_STATEINDICATOR,
"\nBackward Command Received");
        break;
    }
}

int CVICALLBACK UpdateMapCallback (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
    // Send to iPhone
    unsigned char sendData[T_MAX];
    int LaserData[D_MAX];
    int EffectiveRange[RANGE];
    double alpha;
    unsigned char hiByte;
    unsigned char loByte;
    int LockAngle;
    int LockDistance;
    double xLock[2];
    double yLock[2];
    int i;
    Normal_map *Pointer1;
    double Nor_Map2DX[MAP_2D_1DEC_SIZE];
    //Real odometry map
    double Nor_Map2DY[MAP_2D_1DEC_SIZE];
    int *Pointer2;
    switch (event)
    {
        case EVENT_TIMER_TICK:
            Pointer1 = GetPointerToMap2D();
            for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
                Nor_Map2DX[i]=Pointer1 -> Normal_Map2DX[i];
                Nor_Map2DY[i]=Pointer1 -> Normal_Map2DY[i];
            }
            ReleasePointerToMap2D();
            Pointer2 = GetPointerToGoal();
            LockAngle=*Pointer2;
            ReleasePointerToGoal();
            // 1.Constraint LockAngle
            if(LockAngle>171) LockAngle=170;

```

```

        // 2.Add synchronous bytes
        sendData[0]=83;
        sendData[1]=115;
        sendData[T_MAX-2]=69;
        sendData[T_MAX-1]=101;
        //3.Construct sendData from LaserData
        for(i=0;i<D_MAX;i++){
            LaserData[i] =
1000*floor(sqrt(Nor_Map2DX[i]*Nor_Map2DX[i]+Nor_Map2DY[i]*Nor_Map2DY[i]))
;
            hiByte = LaserData[i]/256;
            loByte = LaserData[i]%256;
            sendData[i+2]=hiByte;
            sendData[i+2+D_MAX]=loByte;
        } // End of for

        //4. Calculate LockDistance

        LockDistance = LaserData[LockAngle]; // Initialise the LockDistance
as the safe boundary

        // 5. Reconstruct sendData
        sendData[T_MAX-3]= LockAngle;
        sendData[T_MAX-4]= LockDistance%256; // loByte
        sendData[T_MAX-5]= LockDistance/256; // hiByte
        // 6. Send to iPhone
        if(isConnected){
            ServerTCPWrite (g_hconversation,sendData,T_MAX,TCPTIMEOUT);
        }
        break;
    }
    return 0;
}

// MENU CALLBACK FUNCTIONS
void CVICALLBACK AboutMenuCallback (int menuBar, int menuItem, void
*callbackData,
    int panel)
{
    switch (menuItem)
    {
        {
            /* Take action depending on which item was selected */
            case MENUBAR_HELP_ABOUT:
                DisplayPanel (aboutpanel);
                break;
        }
    }
}

// MENU EXIT CALLBACK FUNCTIONS
void CVICALLBACK MENUEXITCALLBACK (int menuBar, int menuItem, void
*callbackData,
    int panel)
{
    switch (menuItem)
    {
        {
            /* Take action depending on which item was selected */
            case MENUBAR_SYSTEM_EXIT:
                CloseProgram();

```

```

                                QuitUserInterface (0);
                                break;
                                }
}
// About panel callback
int CVICALLBACK ABOKCallBack (int panel, int control, int event,
                                void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            HidePanel (aboutpanel);
            break;
    }
    return 0;
}

```

B.3 Library Code

```

#include <userint.h>
#include <NIDAQmx.h>
#include <rs232.h>
#include <utility.h>
#include <formatio.h>
#include <ansi_c.h>
#include "visa.h"
#include "Library.h"
#include "EncoderDLL.h"
#include "USB.h"

/*****
SEARCHING COM PORTS' FUNCTION
*****/

/* This function will search COM ports to find the ports which are
assigned to URG lasers and Ultrasonic sensors.
input
    char device = "URG" or "USB"
Return
    The Com port's number corresponding to the assigned device.
    -1 will be return if these devices are not found.*/

int SearchCOMPORT_Advanced (char device[25])
{
    ViStatus    status;
    ViSession    defaultRM;
    ViSession    instr;
    ViChar        rsrcName[VI_FIND_BUFLen];
    ViChar        intfDesc[VI_FIND_BUFLen];

```

```

ViUInt32  retCount;
ViFindList flist;
ViUInt16 version;
char *pch;
int COMPORT = -1;

status = viOpenDefaultRM(&defaultRM);
if (status < VI_SUCCESS) {
    /* Error Initializing VISA...exiting */
    return -1;
}
status = viFindRsrc (defaultRM, "ASRL?*INSTR", &flist, &retCount,
rsrcName);
while (retCount--) {
    status = viOpen (defaultRM, rsrcName, VI_NULL, VI_NULL, &instr);
    if (status < VI_SUCCESS) return -1;
    else{
        status = viGetAttribute (instr, VI_ATTR_INTF_INST_NAME,
intfDesc);
        pch = strstr (intfDesc,device);
        if (pch != NULL) {
            status = viGetAttribute (instr, VI_ATTR_INTF_NUM, &version);
            COMPORT = version;
        }
        status = viClose (instr);
    }
    status = viFindNext (flist, rsrcName);
}
viClose (flist);
viClose (defaultRM);
return COMPORT;
}

/* This function will search COM ports to find the port which is assigned
to URG laser.
-1 will be return if URG laser port is not found
NOTE: Make sure that NI-VISA has been installed before use this function
*/

int SearchCOMPORT (int *Laser_port, int *Head_Movement_Port)
{
    ViStatus status;
    ViSession defaultRM;
    ViSession instr;
    ViChar rsrcName[VI_FIND_BUFLen];
    ViChar intfDesc[VI_FIND_BUFLen];
    ViUInt32 retCount;
    ViFindList flist;
    ViUInt16 version;
    char *pch;
    int COMPORT = -1;

    status = viOpenDefaultRM(&defaultRM);
    if (status < VI_SUCCESS) {
        /* Error Initializing VISA...exiting */

```

```

        return -1;
    }
    status = viFindRsrc (defaultRM, "ASRL?*INSTR", &flist, &retCount,
rsrcName);
    while (retCount--) {
        status = viOpen (defaultRM, rsrcName, VI_NULL, VI_NULL, &instr);
        if (status < VI_SUCCESS) return -1;
        else{
            status = viGetAttribute (instr, VI_ATTR_INTF_INST_NAME,
intfDesc);
            pch = strstr (intfDesc,"URG");
            if (pch != NULL) {
                status = viGetAttribute (instr, VI_ATTR_INTF_NUM,
&version);
                *Laser_port = version;
                pch = NULL;
            }
            pch = strstr (intfDesc,"Pro");
            if (pch != NULL) {
                status = viGetAttribute (instr, VI_ATTR_INTF_NUM,
&version);
                *Head_Movement_Port = version;
                pch = NULL;
            }
            status = viClose (instr);
        }
        status = viFindNext (flist, rsrcName);
    }
    viClose (flist);
    viClose (defaultRM);
    return 1;
}

/*****
        LASER URG-04LX' FUNCTION
*****/
/*Init the URG sensor by sending:
Input
    1. Comport: specifies the COM to request data.
    2. Res: resolution = 380/1024*res.
    3. BegAng: Begin scan angle (-30 to 240)
    4. EndAngle: Ending scan angle (-30 to 240)
Output
    1. Return value is size of laser data.
    2. Return command to drive the URG. */
int InitURG (int comport, int res, int BegAng, int EndAng, char *command)
{
    int inqlen;
    char commandtemp[50], temchar[50];;
    //1. Initialising COM port
    OpenComConfig (comport, "", 19200, 0, 8, 1, 2000, 2000);
    //Change baud rate
    sprintf (commandtemp, "S75000000000000\n");
    ComWrt (comport, commandtemp, strlen(commandtemp));
    Delay(0.1);
    CloseCom (comport);
}

```

```

OpenComConfig (comport, "", 750000, 0, 8, 1, 2000, 2000);
//2. Start Laser sensor
BegAng = 128 + (int) (BegAng*1024/360);
sprintf (commandtemp, "G");
Fmt (temchar, "%s<%i", BegAng);
strcat (commandtemp, temchar);
EndAng = 128 + (int) (EndAng*1024/360);
Fmt (temchar, "%s<%i", EndAng);
strcat (commandtemp, temchar);
if(res<10){
    Fmt (temchar, "%s<%i", res);
    strcat (commandtemp, "0");
    strcat (commandtemp, temchar);
}
else{
    Fmt (temchar, "%s<%i", res);
    strcat (commandtemp, temchar);
}
strcat (commandtemp, "\n");
ComWrt (comport, commandtemp, strlen(commandtemp));
Delay(0.1);
sprintf (command, commandtemp);
return ((int) ((EndAng - BegAng)/res)+1);
}

//Collect laser data from port.
//Input:
// 1. Comport: specifies the COM to request data.
// 2. sizeofLdata: size of laser data provided by the InitURG
function.
// 3. Command: Command to drive the URG provided by InitURG function.
// 4. Filter: enable filter for laser data
//Output:
// 1. laserdataX: pointer to X axis laser data
// 2. laserdataY: pointer to Y axis laser data
// 3. Reading status: 1 data is OK, 0 no data.
// 4. Laserdata: polar data for display
int LaserDataDAQ (int comport, int sizeofLdata, double *laserdataX,
double *laserdataY, int res, char *command)
{
    int inqlen, i, count=0;
    char *datatemp;
    double laserdata[180];

    inqlen = GetInQLen (comport);
    if (abs(inqlen - sizeofLdata*2) < 48){
        //1. Read data from port
        datatemp = malloc(inqlen);
        ComRd (comport, datatemp, inqlen);
        CopyString (datatemp, 0, datatemp, 12, inqlen-14);
        FlushInQ (comport);
        FlushOutQ (comport);
        //2. Request new laser data from port
        ComWrt (comport, command, strlen(command));
        //3. Convert data to int type
        for (i=0;i<inqlen;i++)

```

```

        if(datatemp[i] != 10){
            datatemp[count] = datatemp[i] - 48;
            count = count+1;
        }
        for(i=0; i<sizeofLdata; i++){
            laserdata[i] = datatemp[i*2]*64 + datatemp[i*2+1];
            if ((laserdata[i] <= 20)|| (laserdata[i] > 4096))
                laserdata[i] = 4096;
        }
        for(i=0; i<sizeofLdata; i++){
            laserdataX[i] = laserdata[i]*cos(i*2*3.14159/1024*res)/1000;
            laserdataY[i] = laserdata[i]*sin(i*2*3.14159/1024*res)/1000;
        }
        return (1);          //Read data status is OK.
    }
    else
        return (0);          //No data is collected
}

//Close URG sensor operation.
void CloseURG (int comport)
{
    CloseCom(comport);
}

//Filter the noise from the laserdata.
void Laserdatafilter (int sizeofLdata, double *laserdata, double
threshold)
{
    int i;
    for(i=0; i<sizeofLdata; i++)
        if ((i>0)&&(i<sizeofLdata-2)&&(laserdata[i] >=
threshold)&&(laserdata[i+1] >= threshold)&&(laserdata[i-1] <
threshold)&&(laserdata[i+2] < threshold)){
            laserdata[i] = laserdata[i-1];
            laserdata[i+1] = laserdata[i+2];
        }
    else
        if ((i>0)&&(i<sizeofLdata-1)&&(laserdata[i] >=
threshold)&&(laserdata[i+1] < threshold)&&(laserdata[i-1] <
threshold)){
            laserdata[i] = laserdata[i-1];
        }
    else
        if ((i>0)&&(i<sizeofLdata-3)&&(laserdata[i] >=
threshold)&&(laserdata[i+1] >= threshold)&&(laserdata[i+2] >=
threshold)&&(laserdata[i-1] < threshold)&&(laserdata[i+3] <
threshold)){
            laserdata[i] = laserdata[i-1];
            laserdata[i+1] = laserdata[i+3];
            laserdata[i+2] = laserdata[i+3];
        }
    else

```

```

        if ((i>0)&&(i<sizeofLdata-4)&&(laserdata[i-1] <
threshold)&&(laserdata[i] >= threshold)&&(laserdata[i+1] >=
threshold)&&(laserdata[i+2] >= threshold)&&(laserdata[i+3] >=
threshold)&&(laserdata[i+4] < threshold)){
            laserdata[i] = laserdata[i-1];
            laserdata[i+1] = laserdata[i-1];
            laserdata[i+2] = laserdata[i+4];
            laserdata[i+3] = laserdata[i+4];
        }
    }

/*****
        ULTRASONIC SENSOR SF01S' FUNCTION
*****/

/* 1.Initialize COM port for the SF01 and configure setting its
parameters.
Objective: This program open the comport corresponding to the S27-USB-
Serial , then set parameters for the port which include:Baud
rad:9600;Parity: 0; Databits: 8; Stopbit: 1;Size of input queue: 2000;
and Size of output queue: 2000
Input: the number of COM port which represents for S27-USB-Serial
Return: Negative value if an error occurs.
*/
int Init_Config_SF01(int Comport_S27)
{
    return (OpenComConfig (Comport_S27, "", 9600, 0, 8, 1, 2000,
2000));
}

/*
2. This function is in order to command to the SR01 with specific address
to read a range value.
Input:
    Comport_S27:the number of COM port which represents for S27-USB-
Serial
    Address_SF01: the address of the particular sensor ranging from 1
to 16; if the value is out of the range, the address is assigned to
1 by default.
Output:
    Value: which show a distance from an obstacle in the detected cone
shape to the Sensor.
*/
void Read_Range_SF01(int Comport_S27, int Address_SF01,unsigned int
*Value)
{
    char command[2];
    char tempchar [10];
    unsigned char temper [10];
    // Determine the sensor address.
    if ((Address_SF01>0)&&(Address_SF01<17)){
        if (Address_SF01==1) command[0] = 0x01;
        if (Address_SF01==2) command[0] = 0x02;
        if (Address_SF01==3) command[0] = 0x03;
        if (Address_SF01==4) command[0] = 0x04;
        if (Address_SF01==5) command[0] = 0x05;
    }
}

```

```

        if (Address_SF01==6) command[0] = 0x06;
        if (Address_SF01==7) command[0] = 0x07;
        if (Address_SF01==8) command[0] = 0x08;
        if (Address_SF01==9) command[0] = 0x09;
        if (Address_SF01==10) command[0] = 0x10;
        if (Address_SF01==11) command[0] = 0x11;
        if (Address_SF01==12) command[0] = 0x12;
        if (Address_SF01==13) command[0] = 0x13;
        if (Address_SF01==14) command[0] = 0x14;
        if (Address_SF01==15) command[0] = 0x15;
        if (Address_SF01==16) command[0] = 0x16;
    }
    else
        command[0] = 0x01;
    // Set ranging mode into cm
    ComBreak (Comport_S27, 2);
    command[1]= 0x51;
    ComWrt (Comport_S27,command,2);
    // Discard echoed characters from TX
    while (GetInQLen(Comport_S27)<3);
    ComRd(Comport_S27,tempchar,3);
    // SRF01 with address 1
    ComBreak (Comport_S27, 2);
    command[1]= 0x5E;
    ComWrt(Comport_S27,command,2);
    // Discard echoed characters from TX
    while (GetInQLen(Comport_S27)<3);
    ComRd(Comport_S27,tempchar,3);
    //Read range.
    while (GetInQLen(Comport_S27)<2); // wait for range to arrive
    ComRd(Comport_S27,temper,2);
    //Convert bytes into real value and return value.
    *Value =(temper[0]<<8)+temper[1];
}

/*****
ENCODERS' FUNCTION
*****/
// Objective: Initialize USB4 devices and configure setting parameters
for encoders.
void Init_Config_USB4()
{
    short iDeviceCount =0;
    // Initate the USB1
    USB4_Initialize(&iDeviceCount);
    // Configure encoder channel 0.
    USB4_SetPresetValue(0,0,ENCODER_RES-1);
    /* Set the preset register to the CPR-1
    If you plan to select the following counter modes; Range-limit mode, Non-
    recycle mode, or Modulo-N mode (See section 6.1.2); the preset register
    must be set to your desired value. Usually, the preset value is set to
    the encoder's counts per revolution (CPR) minus one.
    */
    USB4_SetMultiplier(0,0,0); // Set quadrature mode to X4.

```

```

/*Select quadrature mode in Control Register (reg. #3)Bit 15 and 14
determine how the encoder counter increments: These bits may be referred
to as either quadrature mode or multiplier.
*/

    USB4_SetCounterMode(0,0,3); // Set counter mode to modulo-N.
// Select count mode in Control Register (reg. #3)
    USB4_SetForward(0,0,1); // Optional: determines the direction of
counting.
    USB4_SetCounterEnabled(0,0,1); // Enable the counter. **IMPORTANT**

    USB4_ResetCount(0,0); // Reset the counter to 0
    // Configure encoder channel 1.
    USB4_SetPresetValue(0,1,ENCODER_RES-1); // Set the preset register
to the CPR-1
    USB4_SetMultiplier(0,1,0); // Set quadrature mode to X4.
    USB4_SetCounterMode(0,1,3); // Set counter mode to modulo-N.
    USB4_SetForward(0,1,1); // Optional: determines the direction of
counting.
    USB4_SetCounterEnabled(0,1,1); // Enable the counter. **IMPORTANT**

    USB4_ResetCount(0,1); // Reset the counter to 1

}

//Compute the movement of the wheelchair between two consecutive
computational cycles.
//Inputs:
//    1. Wheel_Dim: The dimension of the steering wheel
//    2. Axis_Length: The distance between two steering wheels
//Outputs:
//    1. Del_Dis: The straight movement of the wheelchair (central point)
between two computational cycles
//    2. Del_Phi: The turning movement of the wheelchair (central point)
between two computational cycles
void SpeedElementEstimation_USB4(double *Del_Dis, double *Del_Phi,
                                int          Wheel_Dim,      int
Axis_Length)
{
    static unsigned long PrevPos1 =0, PrevPos2 =0;
    unsigned long NewPos1, NewPos2;
    double Factor;
    int Delta1, Delta2;

    // Read the first encoder's data
    USB4_GetCount(0,0,&NewPos1);
    Delta1 = NewPos1- PrevPos1;
    PrevPos1 = NewPos1;
    if (Delta1<-3000){
        Delta1=Delta1+ENCODER_RES;
    }
    if (Delta1>3000){
        Delta1=Delta1-ENCODER_RES;
    }
    // Read the second encoder's data.
    USB4_GetCount(0,1,&NewPos2);

```

```

    Delta2 = NewPos2 - PrevPos2;
    PrevPos2 = NewPos2;
    if (Delta2<-3000){
        Delta2=Delta2+ENCODER_RES;
    }
    if (Delta2>3000){
        Delta2=Delta2-ENCODER_RES;
    }
    // Calculate the straight and turning movement.
    Factor = Wheel_Dim*PI/ENCODER_RES;
    *Del_Dis = (Delta1 + Delta2)*Factor/1000/2; // unit (m)
    *Del_Phi = (Delta2 - Delta1)*Factor/Axis_Length;
}

// Close the connection between the USB4 and its drive.
void Exit_USB4 ()
{
    USB4_Shutdown();
}

/*****
                        HEAD MOVEMENT INTERFACE'S FUNCTION
*****/

/*
2. This function is in order to open the COM port of the head movement
hardware.
Input:
    Comport:the number of COM port which represents for the hardware
*/

void OpenHeadMovementPort(int Comport)
{
    char buffer[16];
    int len;
    double valX, valY;

    OpenComConfig (Comport, "", 115200, 0, 8, 1, 512, 512);
    len = ComRdTerm (Comport, buffer, 15, 13); // 13 = CR byte value
    buffer[len] = 0;
    len = ComRdTerm (Comport, buffer, 15, 13);
    buffer[len] = 0;
    valX = atof (&buffer[1]);
    len = ComRdTerm (Comport, buffer, 15, 13);
    buffer[len] = 0;
    valY = atof (&buffer[1]);
}

int GetRawHeadMovementData(int Comport, int *RLVal, int *FBVal)
{
    char buffer[16];
    int len=0;
    double valX, valY;

    len = ComRdTerm (Comport, buffer, 15, 13); // 13 = CR byte value

```

```

    buffer[len] = 0;
    len = ComRdTerm (Comport, buffer, 15, 13); // 13 = CR byte value
    buffer[len] = 0;
    valX = atof (&buffer[1]);
    *FBVal = RoundRealToNearestInteger(valX);
    len = ComRdTerm (Comport, buffer, 15, 13); // 13 = CR byte value
    buffer[len] = 0;
    valY = atof (&buffer[1]);
    *RLVal = RoundRealToNearestInteger(valY);
    FlushInQ (Comport);
    return 1;
}

int RawHeadDataConvert (int RLraw, int FBraw, int Start_L, int Start_R, int
Start_F, int Start_B, double *Vforward, double *Vright)
{
    double valX, valY;
    valX = FBraw;
    valY = RLraw;
    // turn right signals
    if ((valY>Start_R)&&(valY<Start_L))
        *Vright = 2.5;
    if (valY<=Start_R)
        *Vright = 2.5-0.00015*(valY+500); //for greater than 2.5V ,
turn right
    if (valY>=Start_L)
        *Vright = 2.5-0.00012*(valY+800); // for less than 2.5V ,
turn left
    // Go forward signals
    if ((valX>Start_B)&&(valX<Start_F))
        *Vforward = 2.5;
    if (valX<=Start_B)
        *Vforward = 0.0002*(valX+2190)+2.5;
    if (valX>=Start_F)
        *Vforward = 0.00012*(valX+2000)+2.5; // for greater
than 2.5V , move forward
    return 1;
}

int RawHeadDataFilter (int RLraw, int FBraw, int Thres_Filter,
int *RLraw_F, int *FBraw_F ){
    static int Store_RL,Store_FB;
    static int RL_Slope=0, FB_Slope=0;
    static int St=0;
    int Dif_RL, Dif_FB;
    if (St<1){ // Store data;
        Store_RL=RLraw;
        Store_FB=FBraw;
        St++;
        *RLraw_F = RLraw;
        *FBraw_F = FBraw;
    }
    else{ // Start to filter
        Dif_RL =RLraw-Store_RL;

```

```
Dif_FB =FBraw-Store_FB;
if ((abs(Dif_RL)>Thres_Filter))
    *RLraw_F = Store_RL;//RL_Slope+
else
    *RLraw_F = RLraw;
if ((abs(Dif_FB)>Thres_Filter))
    *FBraw_F = Store_FB; //FB_Slope+
else
    *FBraw_F = FBraw;
FB_Slope = *FBraw_F - Store_FB;
RL_Slope = *RLraw_F - Store_RL;
Store_FB=*FBraw_F;
Store_RL=*RLraw_F;
}

return 1;
}

int GetHeadMovementCommand(int Comport, double *Vforward, double *Vright,
int Start_L,int Start_R, int Start_F, int Start_B )
{
    char buffer[16];
    int len=0;
    double valX, valY;
    len = ComRdTerm (Comport, buffer, 15, 13); // 13 = CR byte value
    buffer[len] = 0;
    len = ComRdTerm (Comport, buffer, 15, 13); // 13 = CR byte value
    buffer[len] = 0;
    valX = atof (&buffer[1]);
    len = ComRdTerm (Comport, buffer, 15, 13); // 13 = CR byte value
    buffer[len] = 0;
    valY = atof (&buffer[1]);
    // turn right signals
    if ((valY>Start_R)&&(valY<Start_L))
        *Vright = 2.5;
    if (valY<=Start_R)
        *Vright = 2.5-0.00015*(valY+500); //for
greater than 2.5V , turn right
    if (valY>=Start_L)
        *Vright = 2.5-0.00012*(valY+800); // for less
than 2.5V , turn left
    // Go forward signals
    if ((valX>Start_B)&&(valX<Start_F))
        *Vforward = 2.5;
    if (valX<=Start_B)
        *Vforward = 0.0002*(valX+2190)+2.5;
    if (valX>=Start_F)
        *Vforward = 0.00012*(valX+2000)+2.5; //
for greater than 2.5V , move forward
    FlushInQ (Comport);
    return 1;
}

void Head_Movement_Filter(double Vforward, double Vright, double
*V_Forward_Filtered,double *V_Right_Filtered){
```

```
float64 dif;
static int Vf=2500, Vr=2500;    // to store the Vforward and Vright
int Fnoise, Rnoise=0;
    //test noise signal
    Fnoise = 0;
    if ((Vforward>3.5)|| (Vforward<1.5))
        Fnoise = 1;
    if (Fnoise ==0){
        *V_Forward_Filtered = Vforward;
        Vf = RoundRealToNearestInteger(Vforward *1000);
    }
    else
    {
        *V_Forward_Filtered = Vf;
        *V_Forward_Filtered=*V_Forward_Filtered/1000;
    }
    Rnoise = 0;
    if ((Vright>3.5)|| (Vright<1.5))
        Rnoise = 1;
    if (Rnoise ==0) {
        *V_Right_Filtered = Vright;
        Vr = RoundRealToNearestInteger(Vright *1000);
    }
    else
    {
        *V_Right_Filtered = Vr;
        *V_Right_Filtered = *V_Right_Filtered/1000;
    }
}

void Head_Movement_Noise_Filter(double Vforward, double Vright, double
*V_Forward_Filtered,double *V_Right_Filtered){
    static int F_Pre=2500,R_Pre= 2500;
    static int F_First=2500,R_First= 2500;
    static int F_Second=2500,R_Second= 2500;
    double F_Slope_Change, R_Slope_Change;
    static int NOISE = 0, K =0;
    const double Noise_Threshold = 0.17;
    double FD, RD;
    FD = Vforward- (double)F_Pre/1000;
    RD = Vright - (double)R_Pre/1000;
    F_Slope_Change = (double)(F_First - F_Second)/1000;
    R_Slope_Change = (double)(R_First - R_Second)/1000;
    if ((fabs(FD)>=Noise_Threshold)|| (fabs(RD)>=Noise_Threshold))
    {
        NOISE = 1;
        K++;
    }
    else {
        NOISE =0;
        K=0;
    }
    if ((NOISE ==1)&&(K%2!=0)) {
        *V_Forward_Filtered= (double)F_Pre/1000+F_Slope_Change;
```

```

        *V_Right_Filtered = (double)R_Pre/1000+R_Slope_Change;
    }
    else {
        *V_Forward_Filtered = Vforward;
        *V_Right_Filtered = Vright;
    }
    F_Second = F_First;
    F_First = *V_Forward_Filtered;
    R_Second = R_First;
    R_First = *V_Right_Filtered;
    R_Pre = RoundRealToNearestInteger(Vright *1000);
    F_Pre = RoundRealToNearestInteger(Vforward *1000);
}

/*****
MAPPING'S FUNCTION
*****/

//Move data to the new position
//Input:
//    1. dX: new position in X axis
//    2. dY: new position in Y axis
//    3. dO: turning angle to new position
//    4. sizeofLdata: size of data
//    5. inputdataX
//    6. inputdataY
//Output:
//    1. outputdataX
//    2. outputdataY
//    3. index of 90o element.
void MoveDataXYO (double dX, double dY, double dO, double *inputdataX,
double *inputdataY, double *outputdataX, double *outputdataY, int
sizeofLdata)
{
    int i;
    double tempX, tempY;

    dX = (double)dX/1000;
    dY = (double)dY/1000;
    for (i = 0; i < sizeofLdata ; i++){
        tempX = (inputdataX[i] - dX)*cos(dO) + (inputdataY[i] -
dY)*sin(dO);
        tempY = -(inputdataX[i] - dX)*sin(dO) + (inputdataY[i] -
dY)*cos(dO);
        outputdataX[i] = tempX;
        outputdataY[i] = tempY;
    }
}

//Reorganise laser data to create the 2D map
//Reform all the data to datares resolution.
//Input:
//    1. dataX: x value of data
//    2. dataY: y value of the data

```

```

//    3. sizeofdata: size of data
//Output:
//    1. polardata: output data in form of polar
void ReformData (const double *dataX, const double *dataY, double
*polardata,int sizeofdata)
{
    double ang, angles, temp;
    int i, j, preindex, index;
    int Index_Max, Index_Min;
    Index_Max = 0;
    Index_Min = 360;
    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        polardata[i] =LASER_RANGE_MAX/1000; // turn into (m) unit
    }
    for (i = 0; i < sizeofdata; i++){
        ang = atan2(dataY[i], dataX[i]);
        if (ang<0) ang = 2*PI + ang;
        index = RoundRealToNearestInteger(ang*180/PI);
        if (index == 360) index = 0;
        if (index < 0) index = 0;
        temp = sqrt(pow(dataX[i],2) + pow(dataY[i],2));
        if (temp > MAP_RANGE_MAX)
            temp = MAP_RANGE_MAX;
        if (polardata[index] > temp)
            polardata[index] = temp;
    }
    Laserdatafilter (MAP_2D_1DEC_SIZE, polardata, LASER_RANGE_MAX);
}

//Reorganise laser data to create the 2D map
//Reform all the data to datares resolution.
//Input:
//    1. dataX: x value of data
//    2. dataY: y value of the data
//    3. sizeofdata: size of data
//Output:
//    1. polardata: output data in form of polar in (m)
//    2. Index_min: starting index of the data corresponding to the begin
//    angle in the reference origin
//    3. Index_Max: ending index of the data corresponding to the end angle
//    in the reference origin

void ReformData1 (const double *dataX, const double *dataY, double
*polardata, int *Index_Min, int *Index_Max,int sizeofdata)
{
    double ang, angles, temp;
    int i, j, preindex, index;
    *Index_Max = 0;
    *Index_Min = 180;
    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        polardata[i] =LASER_RANGE_MAX/1000; // turn into (m) unit
    }
    for (i = 0; i < sizeofdata; i++){
        ang = atan2(dataY[i], dataX[i]);
        if (ang<0) ang = 2*PI + ang;
        index = RoundRealToNearestInteger(ang/(PI/180));

```

```

        if (index >= 360) index = index - 360;
        if (index < 0) index = 0;
        temp = sqrt(pow(dataX[i],2) + pow(dataY[i],2));
        if (temp > MAP_RANGE_MAX)
            temp = MAP_RANGE_MAX;
        if (polardata[index] > temp)
            polardata[index] = temp;
        if (index>*Index_Max) *Index_Max=index;
        if (index<*Index_Min) *Index_Min=index;
    }
    Laserdatafilter (MAP_2D_1DEC_SIZE, polardata, LASER_RANGE_MAX);
}
//Join the new laser data with old 2Dmap to build the new 2D map
void Buildnew2Dmap (double *Lpolar, double *Map2Dpolar, double *Map2DX,
double *Map2DY,int angle_b,int angle_e)
{
    int i;
    double ang;
    int index;
    int a, b;
    a= 0;
    b =180;
    // find the region to start coping the laser data
    for (i = 0; i < 180; i++) {
        if ((Map2DX[i]>0)&&(Map2DY[i]<=0.550)){
            ang = atan2 (Map2DY[i], Map2DX[i]);
            index = RoundRealToNearestInteger(ang/(PI/180));
            if (index>a) a = index;
        }
        if ((Map2DX[i]<0)&&(Map2DY[i]<=0.550)){
            ang = atan2 (Map2DY[i], Map2DX[i]);
            index = RoundRealToNearestInteger(ang/(PI/180));
            if (index<b) b = index;
        }
    }
    if (a>angle_b)
        angle_b = a;
    if (b<angle_e)
        angle_e = b;

    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        if ((i >= angle_b)&&(i <= angle_e))
            if (Map2Dpolar[i]*sin(i*PI/180)>0.55)
                Map2Dpolar[i] = Lpolar[i];
            Map2DX[i] = Map2Dpolar[i]*cos(i*PI/180);
            Map2DY[i] = Map2Dpolar[i]*sin(i*PI/180);
        }
    }
}

//Compensate the wheelchair dimension. Rectangular type.
//Input:
// 1. Map2DX: data of 2D map
// 2. Map2DY: data of 2D map
// 3. leftbound, rightbound, frontbound, rearbound, safewidth,
safelength: dimensions of the wheelchair

```

```

// these dimensions are the boundaries that are accounted from the
// center of the driver wheels
// to the left, right, ... boundaries of the wheelchair. All are in
// mm.
//Output:
// 1. polar2Dmap: the real freespace after the compensating algorithm
// is applied.
void RecDimComp2DMap (const double *Map2DX, const double *Map2DY, double
*polar2Dmap, int leftbound, int rightbound, int frontbound, int
rearbound, int safewidth, int safelength)
{
    int i, j, num, beg, index;
    double aX, aY, bX, bY, cX, cY, dX, dY;
    double ang, angl, ang2;
    double temp1, temp2, range;
    leftbound = leftbound + safewidth;
    rightbound = rightbound + safewidth;
    frontbound = frontbound + safelength;
    rearbound = rearbound + safelength;
    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        //1. Compensate the wheelchair dimension for AB
        aX = Map2DX[i] + (double)(leftbound)/1000;
        aY = Map2DY[i] + (double)(rearbound)/1000;
        angl = atan2(aY,aX);
        if (angl < 0) angl = 2*PI + angl;
        bX = Map2DX[i] - (double)(rightbound)/1000;
        bY = Map2DY[i] + (double)(rearbound)/1000;
        ang2 = atan2(bY,bX);
        if (ang2 < 0) ang2 = 2*PI + ang2;
        //2. Find the begin angle to compensate and number of
        compensated elements
        if ((angl>3*PI/2)&&(ang2<PI/2)){
            num = RoundRealToNearestInteger((2*PI - (angl -
            ang2))/(2*PI/360));
        }
        else
            if ((ang2>3*PI/2)&&(angl<PI/2)){
                num = RoundRealToNearestInteger((2*PI - (ang2 -
                angl))/(2*PI/360));
                angl = ang2;
            }
            else{
                num = RoundRealToNearestInteger((angl -
                ang2)/(2*PI/360));
                num = abs(num);
                if (angl>ang2) angl = ang2;
            }
        beg = RoundRealToNearestInteger(angl/(2*PI/360));
        //3. Compensating method for AB
        for (j = 0; j < num; j++){
            //1. Compute ranges to obstacle
            temp1 = aY/tan(angl + j*(2*PI/360)); //x[j]
            temp2 = sqrt(temp1*temp1 + aY*aY);
            if (j + beg >= 360) index = j + beg - 360;
            else index = j + beg;
            if (temp2<polar2Dmap[index]){

```

```

        polar2Dmap[index] = temp2;
    }
}

//1. Compensate the wheelchair dimension for BC
cX = Map2DX[i] - (double)(rightbound)/1000;
cY = Map2DY[i] - (double)(frontbound)/1000;
ang1 = atan2(cY,cX);
if (ang1 < 0) ang1 = 2*PI + ang1;
bX = Map2DX[i] - (double)(rightbound)/1000;
bY = Map2DY[i] + (double)(rearbound)/1000;
ang2 = atan2(bY,bX);
if (ang2 < 0) ang2 = 2*PI + ang2;
//2. Find the begin angle to compensate and number of
compensated elements
if ((ang1>3*PI/2)&&(ang2<PI/2)){
    num = RoundRealToNearestInteger((2*PI - (ang1 -
ang2))/(2*PI/360));
}
else
    if ((ang2>3*PI/2)&&(ang1<PI/2)){
        num = RoundRealToNearestInteger((2*PI - (ang2 -
ang1))/(2*PI/360));
        ang1 = ang2;
    }
    else{
        num = RoundRealToNearestInteger((ang1 -
ang2)/(2*PI/360));
        num = abs(num);
        if (ang1>ang2) ang1 = ang2;
    }
beg = RoundRealToNearestInteger(ang1/(2*PI/360));
//3. Compensating method for BC
for (j = 0; j < num; j++){
    //1. Compute ranges to obstacle
    temp1 = cX*tan(ang1 + j*(2*PI/360)); //x[j]
    temp2 = sqrt(temp1*temp1 + cX*cX);
    if (j + beg >= 360) index = j + beg - 360;
    else index = j + beg;
    if (temp2<polar2Dmap[index]){
        polar2Dmap[index] = temp2;
    }
}

//1. Compensate the wheelchair dimension for CD
cX = Map2DX[i] - (double)(rightbound)/1000;
cY = Map2DY[i] - (double)(frontbound)/1000;
ang1 = atan2(cY,cX);
if (ang1 < 0) ang1 = 2*PI + ang1;
dX = Map2DX[i] + (double)(leftbound)/1000;
dY = Map2DY[i] - (double)(frontbound)/1000;
ang2 = atan2(dY,dX);
if (ang2 < 0) ang2 = 2*PI + ang2;
//2. Find the begin angle to compensate and number of
compensated element
if ((ang1>3*PI/2)&&(ang2<PI/2)){

```

```

        num = RoundRealToNearestInteger((2*PI - (ang1 -
ang2))/(2*PI/360));
    }
    else
        if ((ang2>3*PI/2)&&(ang1<PI/2)){
            num = RoundRealToNearestInteger((2*PI - (ang2 -
ang1))/(2*PI/360));
            ang1 = ang2;
        }
        else{
            num = RoundRealToNearestInteger((ang1 -
ang2)/(2*PI/360));
            num = abs(num);
            if (ang1>ang2) ang1 = ang2;
        }
    beg = RoundRealToNearestInteger(ang1/(2*PI/360));
    //3. Compensating method for CD
    for (j = 0; j < num; j++){
        //1. Compute ranges to obstacle
        temp1 = cY/tan(ang1 + j*(2*PI/360)); //x[j]
        temp2 = sqrt(temp1*temp1 + cY*cY);
        if (j + beg >= 360) index = j + beg - 360;
        else index = j + beg;
        if (temp2<polar2Dmap[index]){
            polar2Dmap[index] = temp2;
        }
    }

    //1. Compensate the wheelchair dimension for DA
    aX = Map2DX[i] + (double)(leftbound)/1000;
    aY = Map2DY[i] + (double)(rearbound)/1000;
    ang1 = atan2(aY,aX);
    if (ang1 < 0) ang1 = 2*PI + ang1;
    dX = Map2DX[i] + (double)(leftbound)/1000;
    dY = Map2DY[i] - (double)(frontbound)/1000;
    ang2 = atan2(dY,dX);
    if (ang2 < 0) ang2 = 2*PI + ang2;
    //2. Find the begin angle to compensate and number of
compensated element
    if ((ang1>3*PI/2)&&(ang2<PI/2)){
        num = RoundRealToNearestInteger((2*PI - (ang1 -
ang2))/(2*PI/360));
    }
    else
        if ((ang2>3*PI/2)&&(ang1<PI/2)){
            num = RoundRealToNearestInteger((2*PI - (ang2 -
ang1))/(2*PI/360));
            ang1 = ang2;
        }
        else{
            num = RoundRealToNearestInteger((ang1 -
ang2)/(2*PI/360));
            num = abs(num);
            if (ang1>ang2) ang1 = ang2;
        }
    }
    beg = RoundRealToNearestInteger(ang1/(2*PI/360));

```

```

//3. Compensating method for DA
for (j = 0; j < num; j++){
    //1. Compute ranges to obstacle
    temp1 = aX*tan(ang1 + j*(2*PI/360)); //x[j]
    temp2 = sqrt(temp1*temp1 + aX*aX);
    if (j + beg >= 360) index = j + beg - 360;
    else index = j + beg;
    if (temp2 < polar2Dmap[index]){
        polar2Dmap[index] = temp2;
    }
}

}

}

//Compensate the wheelchair dimension. Circular type.
//Input:
// 1. Map2DX: data of 2D map
// 2. Map2DY: data of 2D map
// 3. centerX, centerY, radius, safearea: dimensions of the wheelchair
// these dimensions are the boundaries that are accounted from the
// center of the driver wheels
// to the left, right, ... boundaries of the wheelchair. All are in
// mm.
//Output:
// 1. polar2Dmap: the real freespace after the compensating algorithm
// is applied.
void CirDimComp2DMap (const double *Map2DX, const double *Map2DY, double
*polar2Dmap, int centerX, int centerY, int radius, int safearea)
{
    int i, j, num, index;
    double tempX, tempY, range, phi;
    double rangetemp, phitemp1, phitemp2, radtemp;
    radtemp = (double)(radius + safearea)/1000;
    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        tempX = Map2DX[i] - (double)(centerX)/1000;
        tempY = Map2DY[i] - (double)(centerY)/1000;
        range = sqrt(tempX*tempX + tempY*tempY);
        phi = atan2(tempY, tempX);
        if (phi < 0) phi = 2*PI + phi;
        if (radtemp > range) range = radtemp;
        num = RoundRealToNearestInteger(asin(radtemp/range)/(2*PI/360));
        for (j = 0; j < num; j++){
            rangetemp = range*cos(j*(2*PI/360)) -
            sqrt(pow(radtemp,2) - pow(range*sin(j*(2*PI/360)),2));
            phitemp1 = phi + j*(2*PI/360);
            phitemp2 = phi - j*(2*PI/360);
            if (phitemp1 >= 2*PI) phitemp1 = phitemp1 - 2*PI;
            if (phitemp1 < 0) phitemp1 = phitemp1 + 2*PI;
            if (phitemp2 >= 2*PI) phitemp2 = phitemp2 - 2*PI;
            if (phitemp2 < 0) phitemp2 = phitemp2 + 2*PI;
            index = RoundRealToNearestInteger(phitemp1/(2*PI/360));
            if (index >= 360) index = index - 360;
            if (index < 0) index = 0;
        }
    }
}

```

```

        if (polar2Dmap[index] > rangetemp) polar2Dmap[index] =
        rangetemp;
        index = RoundRealToNearestInteger(phitemp2/(2*PI/360));
        if (index >=360) index = index - 360;
        if (index < 0) index = 0;
        if (polar2Dmap[index] > rangetemp) polar2Dmap[index] =
        rangetemp;
    }
}

//Obstacle density
//Input:
//    1. polardata
//    2. sizeofdata: size of polar data
//Output:
//    1. densFW: density of obstacle in front of the robot (60-120 deg)
//    2. densR: density of obstacle in the Right of the robot (0-60deg)
//    3. densL: density of obstacle in the Left of the robot (120-180deg)
void ObstacleDensity (double *polardata, int sizeofdata, double
*densFW,double *densR, double *densL, double *maxval){
    int i;
    *densFW = 0;
    *densR = 0;
    *densL = 0;
    *maxval = 0;
    for (i = 0; i < sizeofdata; i++){
        if ((i>= 0)&&(i<60)) *densR = *densR + polardata[i];
        if ((i>=60)&&(i<120)) *densFW = *densFW + polardata[i];
        if ((i>=120)&&(i<180)) *densL = *densL + polardata[i];
        if (polardata[i] > *maxval) *maxval = polardata[i];
    }
}

//Environment Operating Mode
//Input:
//    1. polardata : real-time map
//    2. Current_Direction: the user's current navigation direction (in
degree) given by the interface
//Output:
//    1. Operating mode: density of obstacle in front of the robot (60-
120 deg)

void Environment_Operating_Mode (double *polardata,int Current_Direction,
int *Operating_Mode){
    int i;
    int Region_A, Region_B;
    double Region_A_Threshold,Region_B_Threshold;
    Region_A=0; Region_B=0;
    Region_A_Threshold = 0.5;
    Region_B_Threshold = 1;
    for (i = 0; i < MAP_2D_1DEC_SIZE; i++){
        if (polardata[i]<=Region_A_Threshold)
            Region_A = 1;
        if (polardata[i]<=Region_B_Threshold)
            Region_B = 1;
    }
}

```

```

    }
    if (Region_A)
        Region_B = 0;
    if (Region_A || Region_B){
        if (Region_A){
            *Operating_Mode = MACHINE;
            for (i = 0; i < 6; i++){
                if((Current_Direction-3+i)>=0)
                    if (polardata[(Current_Direction-3+i)]<Region_A_Threshold)
                        *Operating_Mode = URGENT;
            }
        }
        else{
            *Operating_Mode = MANUAL;
            for (i = 0; i < 6; i++){
                if((Current_Direction-3+i)>=0){
                    if (polardata[(Current_Direction-3+i)]<Region_B_Threshold)
                        *Operating_Mode = MACHINE;
                }
            }
        }
    }
    else
        *Operating_Mode = MANUAL;
}

//Find FSs by using threshold
//Input:
// 1. datapolar: the laser data
// 2. threshold: threshold to determine FSs
//Output:
// 1. freespace:
// 2. return: number of FSs
int ThresholdFreespace (double *datapolar, FreeSpace *freespace, double
threshold){
    int i,j, rem = 0, count = 0;
    double densFS = 0;
    for (i = 1; i < MAP_2D_1DEC_SIZE/2; i++){
        if ((rem == 0)&&(datapolar[i] >= threshold)){
            freespace[count].FS_B = i;
            rem = 1;
        }
        if ((rem == 1)&&((datapolar[i] <= threshold)|| (i ==
MAP_2D_1DEC_SIZE/2 - 1))){
            densFS = 0;
            for (j = freespace[count].FS_B; j <= i; j++)
                densFS = densFS + datapolar[j];
            densFS = densFS/(i - freespace[count].FS_B);
            if ((i - freespace[count].FS_B >=
FS_MIN_SIZE)/*&&(densFS >= 0.2)*/){
                freespace[count].FS_E = i;
                count++;
            }
            rem = 0;
        }
    }
}

```

```

    }
}
if (count == 0){
    freespace[count].FS_B = 0;
    freespace[count].FS_E = MAP_2D_1DEC_SIZE/2;
    count++;
}
//2. categorise these possible directions
// - If size <= 60: Small FS
// - If size > 60: Large FS
for (i = 0; i < count; i++){
    if (FSspace[i].FS_E - FSspace[i].FS_B <= 60)
        FSspace[i].typeFS = SMALL;
    else
        FSspace[i].typeFS = LARGE;
}
return count;
}

//UpdateTargetAuto: Update target automatically based on the current
//target
//and current situation (environment). At the very beginning the target
//is
//90 degree and then update to the suitable direction.
//Input:
// 1. preFS_B: the value of begin of free space at previous step
// 2. preFS_E: the value of end of the free space at previous step
// 3. FSspace: the possible directions
// 4. numofFS: the number of FSs
// 5. curpose: the current pose of the WC that is used update new
//target
//Output:
// 1. target: new target for WC
// 2. index: index of active FS.
int UpdateTargetAuto (int preFS_B, int preFS_E, int numofFS, double k, int
*target, FreeSpace *FSspace)
{
    int i, costval, temptar, temptar2, tempcost, index;

    //1. Find the FS which has the smallest value of const function
    index = 0;
    if (FSspace[index].FS_E - FSspace[index].FS_B <= 60){ //Small FS
        temptar = (int)((FSspace[index].FS_E +
FSspace[index].FS_B)/2);
        costval = abs(FSspace[index].FS_B - preFS_B) +
abs(FSspace[index].FS_E - preFS_E) + k*abs(temptar - *target);
    }
    else{ //Large FS
        temptar = FSspace[index].FS_E - EDGE_FS_VAL;
        costval = abs(FSspace[index].FS_B - preFS_B) +
abs(FSspace[index].FS_E - preFS_E) + k*abs(temptar - *target);
        temptar = FSspace[index].FS_B + EDGE_FS_VAL;
        tempcost = abs(FSspace[index].FS_B - preFS_B) +
abs(FSspace[index].FS_E - preFS_E) + k*abs(temptar - *target);
        if (tempcost < costval)
            costval = tempcost;
    }
}

```

```

    for (i = 1; i < numofFS; i++){
        if (FSspace[i].FS_E - FSspace[i].FS_B <= 60){ //Small FS
            temptar = (int)((FSspace[i].FS_E + FSspace[i].FS_B)/2);
            tempcost = abs(FSspace[i].FS_B - preFS_B) +
abs(FSspace[i].FS_E - preFS_E) + k*abs(temptar - *target);
        }
        else{ //Large FS
            temptar = FSspace[i].FS_E - EDGE_FS_VAL;
            tempcost = abs(FSspace[i].FS_B - preFS_B) +
abs(FSspace[i].FS_E - preFS_E) + k*abs(temptar - *target);
            temptar = FSspace[i].FS_B + EDGE_FS_VAL;
            temptar = abs(FSspace[i].FS_B - preFS_B) +
abs(FSspace[i].FS_E - preFS_E) + k*abs(temptar - *target);
            if (temptar < tempcost)
                tempcost = temptar;
        }
        if (costval > tempcost){
            index = i;
            costval = tempcost;
        }
    }

    //2. Update new target
    // curpose = (int)(curpose*180/PI);
    if (FSspace[index].FS_E - FSspace[index].FS_B <= 60) //Small FS
        *target = (int)((FSspace[index].FS_E +
FSspace[index].FS_B)/2);
    else{ //Large FS
        temptar = FSspace[index].FS_E - EDGE_FS_VAL;
        tempcost = abs(FSspace[index].FS_B - preFS_B) +
abs(FSspace[index].FS_E - preFS_E) + k*abs(temptar - *target);
        temptar = FSspace[index].FS_B + EDGE_FS_VAL;
        costval = abs(FSspace[index].FS_B - preFS_B) +
abs(FSspace[index].FS_E - preFS_E) + k*abs(temptar - *target);
        if (costval < tempcost)
            temptar = FSspace[index].FS_B + EDGE_FS_VAL;
        else{
            temptar = FSspace[index].FS_E - EDGE_FS_VAL;
            costval = tempcost;
        }
        temptar2 = (FSspace[index].FS_E + FSspace[index].FS_B)/2;
        tempcost = abs(FSspace[index].FS_B - preFS_B) +
abs(FSspace[index].FS_E - preFS_E) + k*abs(temptar2 - *target);
        if (costval > tempcost)
            *target = temptar2;
        else
            *target = temptar;
        // *target = *target + curpose;
    }

    return index;
}

/*****
NEURAL NETWORK'S FUNCTION

```

```

*****/

//Function_LoadFFNNetPara: load network weight from text file
void Function_LoadFFNNetPara (FFNNNetStruct *netstruct){
    char fDataname[100];
    char buffer[11];
    FILE *fData;
    int i, j, k;
    //Load parameter from the text file
    strcpy (fDataname, "GOA06NNweight.txt");
    fData = fopen (fDataname, "r");      // r: Opens a file for reading.
    //1. Load InputWeight values
    fread (buffer, 1, 7, fData);
    for (i = 0; i < HID_NUM; i++){
        for (j = 0; j < IN_NUM; j++){
            fread (buffer, 1, 11, fData);
            CopyString(buffer, 0, buffer, 0, 10);      //Add the
NULL to the end of the string
            Scan (buffer, "%s>%f", &netstruct->InputWeight[i*IN_NUM
+ j]);      //convert to double
        }
        fread (buffer, 1, 1, fData);
    }

    //2. Load InputBias values
    fread (buffer, 1, 11, fData);
    for (i = 0; i < HID_NUM; i++){
        fread (buffer, 1, 11, fData);
        CopyString(buffer, 0, buffer, 0, 10);      //Add the NULL to
the end of the string
        Scan (buffer, "%s>%f", &netstruct->InputBias[i]);
        //convert to double
    }
    fread (buffer, 1, 1, fData);

    //3. load LayerWeight values
    fread (buffer, 1, 10, fData);
    for (i = 0; i < OUT_NUM; i++){
        for (j = 0; j < HID_NUM; j++){
            fread (buffer, 1, 11, fData);
            CopyString(buffer, 0, buffer, 0, 10);      //Add the
NULL to the end of the string
            Scan (buffer, "%s>%f", &netstruct-
>LayerWeight[i*HID_NUM + j]); //convert to double
        }
        fread (buffer, 1, 1, fData);
    }

    //4. load layerBias values
    fread (buffer, 1, 10, fData);
    for (i = 0; i < OUT_NUM; i++){
        fread (buffer, 1, 11, fData);
        CopyString(buffer, 0, buffer, 0, 10);      //Add the NULL to
the end of the string
        Scan (buffer, "%s>%f", &netstruct->LayerBias[i]);
        //convert to double
    }
}

```

```

        fread (buffer, 1, 1, fData);
    }

void Function_GetNNetInput (double *netinput, double *polardata, double
density, double target){
    int i, count, DAQmode = 0;
    double kzero = 0.01, mean;
    double kforce = 1.2, ktargget = 6;
    double genspeed, rforceX, rforceY, aforceX, aforceY;
    //1. Laser data
    mean = 0; count = 1;
    for (i = 0; i <= MAP_2D_1DEC_SIZE/2; i++){
        if (i < 5*count)
            mean = mean + polardata[i];
        else{
            netinput[count-1] = mean/5;
            count++;
            mean = 0;
        }
    }
    //2. Driving force.
    if (density < 100)        genspeed = MAX_SPEED*0.6*density/100;
    else
        genspeed = MAX_SPEED*0.4*(density - 100)/(320-100) +
MAX_SPEED*0.6;
    rforceX = 0; rforceY = 0;
    for (i = 0; i < MAP_2D_1DEC_SIZE/2; i++){
        rforceX = rforceX + (kforce/pow(polardata[i] + kzero,
1))*cos(i*2*PI/360);
        rforceY = rforceY + (kforce/pow(polardata[i] + kzero,
1))*sin(i*2*PI/360);
    }
    rforceX = kforce*rforceX/MAP_2D_1DEC_SIZE;
    rforceY = kforce*rforceY/MAP_2D_1DEC_SIZE;
    aforceX = -ktargget*genspeed*cos(target*PI/180);
    aforceY = -ktargget*genspeed*sin(target*PI/180);
    netinput[36] = density/100;
    netinput[37] = target*PI/180;
    netinput[38] = rforceX;
    netinput[39] = rforceY;
    netinput[40] = aforceX;
    netinput[41] = aforceY;
}
//Function_CompOutFFNNet: compute output of nn

void Function_CompOutFFNNet(double *netinput, double *netoutput,
FFNNetStruct *netstruct)
{
    int i,j;
    double hidden[HID_NUM];

    //1. Compute output network
    for (i = 0; i < HID_NUM; i++){
        hidden[i] = 0;
        for (j = 0; j < IN_NUM; j++){

```

```

        hidden[i] = hidden[i] + netstruct->InputWeight[i*IN_NUM
+ j]*netinput[j];
    }
    hidden[i] = hidden[i] + netstruct->InputBias[i];
}
//tagsin function
for (i = 0; i < HID_NUM; i++)
    hidden[i] = 2/(1+exp(-2*hidden[i]))-1;
//Linear function
for (i = 0; i < OUT_NUM; i++){
    netoutput[i] = 0;
    for (j = 0; j < HID_NUM; j++)
        netoutput[i] = netoutput[i] + netstruct-
>LayerWeight[i*HID_NUM + j]*hidden[j];
    netoutput[i] = netoutput[i] + netstruct->LayerBias[i];
}
}

/*****
        REMOTE MASTER JOYSTICK CONTROL'S FUNCTION
*****/

//GetJoystickSignal: collect the control data from Joystick
//Input
// 1. Forward_task: virtual forward task
// 2. Right_task
//Output:
// 1. forsignal: forward signal
// 2. rigsignal: turn right signal

void GetJoystickSignal (TaskHandle forward_task, TaskHandle right_task,
float64 *forsignal,float64 *rigsignal)
{
    DAQmxReadAnalogF64 (forward_task, 1, 10.0, DAQmx_Val_GroupByChannel,
forsignal, 1, NULL, NULL);
    DAQmxReadAnalogF64 (right_task, 1, 10.0, DAQmx_Val_GroupByChannel,
rigsignal, 1, NULL, NULL);
}
int ManualControlJoyS (TaskHandle velocitytask, TaskHandle
steeringtask,double velocity, double steering){
    DAQmxWriteAnalogF64 (velocitytask, 1, 1, 10.0, DAQmx_Val_GroupByChannel,
&velocity, NULL, NULL);
    DAQmxWriteAnalogF64 (steeringtask, 1, 1, 10.0, DAQmx_Val_GroupByChannel,
&steering, NULL, NULL);
    return 1; //finish function
}

//Control motor: This function will convert the ref speeds (velocity m/s,
//and steering rad/s) to volts respectively. This process is based on the
//characteristic of the speeds read by encoders and volts that set by
USB6008.
//Input:
// 1. Velocity: the ref velocity [0 -> 1] m/s
// 2. Steering: the ref steering [0 -> 90] dec/s for the turn right
command

```

```

//                                     [-90 -> 0] dec/s for
the turn left command
//Output NULL
int MotorControl (TaskHandle velocitytask, TaskHandle steeringtask, double
velocity, double steering){
    double controlval;
    if (velocity > 0){
        controlval = velocity + 2.53;
        if (controlval < 2.5) controlval = 2.5;
        if (controlval > 3.5) controlval = 3.5;
    }
    else{
        controlval = 1/0.85*velocity + 2.4;
        if (controlval < 1.5) controlval = 1.5;
        if (controlval > 2.5) controlval = 2.5;
    }
    if (velocity == 0) controlval = 2.5;
    DAQmxWriteAnalogF64      (velocitytask,      1,      1,      10.0,
DAQmx_Val_GroupByChannel, &controlval, NULL, NULL);
    controlval = steering*PI/180;
    if (steering > 0){
        controlval = 1/1.9*controlval + 2.50;
        if (controlval < 2.5) controlval = 2.5;
        if (controlval > 3.5) controlval = 3.5;
    }
    else{
        controlval = 1/1.8*controlval + 2.45;
        if (controlval < 1.5) controlval = 1.5;
        if (controlval > 2.5) controlval = 2.5;
    }
    if (steering == 0) controlval = 2.5;
    DAQmxWriteAnalogF64      (steeringtask,      1,      1,      10.0,
DAQmx_Val_GroupByChannel, &controlval, NULL, NULL);
    return 1;    //finish function
}

```

Appendix C

Questionnaire for the evaluation experiments

User Evaluation Questionnaire with The Intelligent Wheelchair Adaptations

Technology device	SAM (Semi-Autonomous Machine)
User Name	
Date of assessment	

The purpose of this questionnaire is to rate your experiences while you are with each of the control modes provided by our intelligent wheelchair system via the head movement interface (HMI). Two control modes include:

- Manual Control (HMI only): You just use the interface to manually drive the wheelchair without any support.
- Semi-autonomous Control (HMI+SAM): You and the machine evolve into the navigation process. Depending on particular situations, the wheelchair will decide whether it should provide its support or not.

There are six questions related to the above modes. Do not leave any question unanswered and please make any comment if possible.

Question 3: How hurried or rushed was the pace of tasks? How much time pressure did you feel due to the pace at which the tasks or task elements occurred?

1	2	3	4	5
Very low	Low	Normal	High	Very high

Manual mode

1 2 3 4 5

Semi-autonomous mode

1 2 3 4 5

Comment:

Question 4: How successful were you in performing the tasks? How satisfied were you with your performance?

1	2	3	4	5
Very low	Low	Normal	High	Very high

Manual mode

1 2 3 4 5

Semi-autonomous mode

1 2 3 4 5

Comment:

Question 5: How insecure, discouraged, irritated, stressed, and annoyed did you feel during the tasks?

1	2	3	4	5
Very low	Low	Normal	High	Very high

Manual mode	1	2	3	4	5
-------------	---	---	---	---	---

Semi-autonomous mode	1	2	3	4	5
----------------------	---	---	---	---	---

Comment:

Question 6: How hard did you have to work (mentally and physically) to accomplish your level of performance?

1	2	3	4	5
Very low	Low	Normal	High	Very high

Manual mode	1	2	3	4	5
-------------	---	---	---	---	---

Semi-autonomous mode	1	2	3	4	5
----------------------	---	---	---	---	---

Comment:

Thank you very much

Bibliography

T.R. A. Lankenau,(2000), 'The Role of Shared Control in Service Robots - The Bremen Autonomous Wheelchair as an Example', *In T. Röfer, A. Lankenau, R. Moratz (Eds.), Service Robotics - Applications and Safety Issues in an Emerging Market. Workshop Notes*, pp. 27–31.

T.R. A. Lankenau,(2001), 'A Safe and Versatile Mobility Assistant', *In Reinventing the Wheelchair. IEEE Robotics and Automation Magazine*, No. 7, pp. 29–37.

T.R. A. Lankenau,(2002), 'Mobile Robot Self-Localization in Large-Scale Environments', *In Proceedings of the IEEE International Conference on Robotics and Automation 2002 (ICRA-2002)*, pp. 1359–1364. *IEEE*.

T.R. A. Lankenau, B. Krieg-Brückner,(2003), 'Self-Localization in Large-Scale Environments for the Bremen Autonomous Wheelchair', *In C. Freksa, W. Brauer, C. Habel, K. F. Wender (Eds.), Spatial Cognition III, Vol. 2685*, pp. 34–61, *Lecture Notes in Artificial Intelligence*.

A.B.o.S.-. ABS,(2010), 'Disability, Ageing and Carers, Australia: Summary of findings 2009', *Cat. No. 4430.0 Canberra*.

Y. Adachi, K. Goto, A. Khiat, Y. Matsumoto & T. Ogasawara,(2003), 'Estimation of User's Attention Based on Gaze and Environment Measurements for Robotic Wheelchair', *Robot*

and Human Interactive Communication, 2003. Proceedings. ROMAN 2003. The 12th IEEE International Workshop on, pp. 97-102.

Y. Adachi, H. Tsunenari, Y. Matsumoto & T. Ogasawara,(2004), 'Guide Robot's Navigation Based on Attention Estimation Using Gaze Information', *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1, pp. 540-5 vol.1.

A.A. Argyros, P. Georgiadis, P. Trahanias & D.P. Tsakiris,(2002), 'Semi-autonomous Navigation of a Robotic Wheelchair', *Journal of Intelligent and Robotic Systems*, 34 (3):315 - 329, 2002.

G. Bebis, R. Boyle, B. Parvin, D. Koracin, S. Wang, K. Kyungnam, B. Benes, K. Moreland, C. Borst, S. DiVerdi, C. Yi-Jen, J. Ming, R. Tomari, Y. Kobayashi & Y. Kuno 2011, 'Multi-view Head Detection and Tracking with Long Range Capability for Social Navigation Planning', *Advances in Visual Computing*, vol. 6939, Springer Berlin Heidelberg, pp. 418-27.

C.M. Bishop,(1995), 'Neural Networks for Pattern Recognition', *Oxford University Press, Inc.* .

J. Borenstein & Y. Koren,(1991), 'Histogramic In-Motion Mapping for Mobile Robot Obstacle Avoidance', *IEEE Transactions on Robotics and Automation*, vol. 7, no. 4, pp. 535-9.

G. Bourhis & Y. Agostini,(1998), 'Man-machine Cooperation for the Control of an Intelligent Powered Wheelchair', *Journal of Intelligent and Robotic Systems*, vol. 22, no. 3-4, pp. 269-87.

M. W. Brault,(2012), 'Americans with Disabilities: 2010', *U.S. Census Bureau*.

C. Butler,(1986), 'Effects of Powered Mobility on Self - Initiated Behaviours of Very Young Children with Locomotor Disability', *Developmental Medicine and Child Neurology*, vol. 28, no. 3, pp. 325-32.

K.H. C. Mandel, T. Vierhuff,(2005), 'Towards an Autonomous Wheelchair: Cognitive Aspects in Service Robotics', *In Proceedings of Towards Autonomous Robotic Systems (TAROS 2005)*, pp. 165–172.

T. Carlson & Y. Demiris,(2012), 'Collaborative Control for a Robotic Wheelchair: Evaluation of Performance, Attention, and Workload', *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 3, pp. 876-88.

H.N. Chow & X. Yangsheng,(2006), 'Learning Human Navigational Skill for Smart Wheelchair in a Static Cluttered Route', *Industrial Electronics, IEEE Transactions on*, vol. 53, no. 4, pp. 1350-61.

G. Del Castillo, S. Skaar, A. Cardenas & L. Fehr,(2006), 'A Sonar Approach to Obstacle Detection for a Vision-Based Autonomous Wheelchair', *Robotics and Autonomous Systems*, vol. 54, no. 12, pp. 967-81.

E. Demeester, M. Nuttin, D. Vanhooydonck, G. Vanacker & H. Van Brussel,(2005), 'Global Dynamic Window Approach for Holonomic and Non-Holonomic Mobile Robots with Arbitrary Cross-Section', *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 2357-62.

K. Edwards & A. McCluskey,(2010), 'A Survey of Adult power Wheelchair and Scooter Users', *Disability and rehabilitation. Assistive technology*, vol. 5, no. 6, pp. 411-9.

- L. Fehr, W.E. Langbein & S.B. Skaar,(2000), 'Adequacy of Power Wheelchair Control Interfaces for Persons with Severe Disabilities: A Clinical Survey', *Journal of Rehabilitation Research and Development*, vol. 37, no. 3, pp. 353-60.
- D. Fox, W. Burgard & S. Thrun,(1997), 'The Dynamic Window Approach to Collision Avoidance', *Ieee Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23-33.
- C. Galindo, J. Gonzalez & J.A. Fernandez-Madrigal,(2006a), 'Control Architecture for Human-Robot Integration: Application to a Robotic Wheelchair', *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 36, no. 5, pp. 1053-67.
- C. Galindo, J. Gonzalez & J.A. Fernandez-Madrigal,(2006b), 'Control Architecture for Human - Robot Integration: Application to a Robotic Wheelchair', *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 36, no. 5, pp. 1053-67.
- T. Hamagami & H. Hirata,(2004), 'Development of Intelligent Wheelchair Acquiring Autonomous, Cooperative, and Collaborative Behavior', *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 4, pp. 3525-30 vol.4.
- S.G. Hart,(2006), 'NASA-task load index (NASA-TLX); 20 years later', *Human Factors and Ergonomics Society Annual Meeting Proceedings* vol. 50, pp. 904–8
- S. Hemachandra, T. Kollar, N. Roy & S. Teller,(2011), 'Following and Interpreting Narrated Guided Tours', *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 2574-9.
- L. Hokuyo Automatic Co.,(2005), 'URG-04LX Product specification', *Technical Document*.

C. Hon Nin, X. Yangsheng & S.K. Tso,(2002), 'Learning Human Navigational Skill for Smart Wheelchair', *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 1, pp. 996-1001 vol.1.

A. Huentemann, E. Demeester, G. Vanacker, D. Vanhooydonck, J. Philips, H. Van Brussel, M. Nuttin & Ieee 2007, *Bayesian Plan Recognition and Shared Control under Uncertainty: Assisting Wheelchair Drivers by Tracking Fine Motion Paths*.

T.N. Instruments,(2012), 'NI USB-6008/6009 User Guide and Specifications', *Bus-Powered Multifunction DAQ USB Device*.

I. Iturrate, J.M. Antelis, A. Kubler & J. Minguéz,(2009), 'A Noninvasive Brain-Actuated Wheelchair Based on a P300 Neurophysiological Protocol and Automated Navigation', *Robotics, IEEE Transactions on*, vol. 25, no. 3, pp. 614-27.

A.L. J. Kollmann , A. Bühlmeier , B. Krieg-Brückner , T. Röfer , B. Krieg--br Uckner , T. R Ofer,(1997), 'Navigation of a Kinematically Restricted Wheelchair by the Parti-Game Algorithm', *In Spatial Reasoning in Mobile Robots and Animals, AISB-97 Workshop*, pp. 35–44. *Manchester University*.

O. Khatib,(1985), 'Real-time Obstacle Avoidance for Manipulators and Mobile Robots', *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2, pp. 500-5.

K. Kidono, J. Miura & Y. Shirai,(2002), 'Autonomous Visual Navigation of a Mobile Robot Using a Human-Guided Experience', *Robotics and Autonomous Systems*, vol. 40, no. 2-3, pp. 121-30.

T. Kruse, A.K. Pandey, R. Alami & A. Kirsch,(2013), 'Human-Aware Robot Navigation: A Survey', *Robotics and Autonomous Systems*, no. 0.

Y. Kuno, N. Shimada & Y. Shirai,(2003), 'Look Where You're Going [Robotic Wheelchair]', *Robotics & Automation Magazine, IEEE*, vol. 10, no. 1, pp. 26-34.

R. Kurozumi, K. Tsuji, S.i. Ito, K. Sato, S. Fujisawa & T. Yamamoto,(2010), 'Experimental Validation of an Online Adaptive and Learning Obstacle Avoiding Support System for the Electric Wheelchairs', *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pp. 92-9.

S.P. Levine, D.A. Bell, L.A. Jaros, R.C. Simpson, Y. Koren & J. Borenstein,(1999), 'The NavChair Assistive Wheelchair Navigation System', *IEEE transactions on rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 7, no. 4, pp. 443-51.

A.C. Lopes, U. Nunes & L. Vaz,(2010), 'Assisted Navigation Based on Shared-control, Using Discrete and Sparse Human-Machine Interfaces', *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pp. 471-4.

E.F. LoPresti, V. Sharma, R.C. Simpson & L.C. Mostowy,(2011), 'Performance Testing of Collision-Avoidance System for Power Wheelchairs', *Journal of Rehabilitation Research and Development*, vol. 48, no. 5, pp. 529-43.

T. Lozano-Perez,(1983), 'Spatial Planning: A Configuration Space Approach', *Computers, IEEE Transactions on*, vol. C-32, no. 2, pp. 108-20.

D.J.C. Mackay,(1992), 'Bayesian Interpolation', *Neural Computation*, vol. 4, no. 3, pp. 415-47.

D.J.C. Mackay,(1992d), 'A Practical Bayesian Framework for Backpropagation Networks', *Neural Computation*, vol. 4, no. 3, pp. 448-72.

C. Mandel, T. Luth, T. Laue, T. Rofer, A. Graser & B. Krieg-Bruckner,(2009), 'Navigating a Smart Wheelchair with a Brain-Computer Interface Interpreting Steady-State Visual Evoked Potentials', *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 1118-25.

R.D.R.a.R.J. Mark,(1999), '*Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*', *The MIT Press, 1999*

Y. Matsumoto, K. Ikeda, M. Inaba & H. Inoue,(2000), 'Exploration and Navigation in Corridor Environment Based on Omni-View Sequence', *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 2, pp. 1505-10 vol.2.

Y. Matsumoto, T. Ogasawara & A. Zelinsky,(2000), 'Behavior Recognition Based on Head Pose and Gaze Direction Measurement', *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 3, pp. 2127-32 vol.3.

M. Mazo,(2001), 'An Integral System for Assisted Mobility [Automated Wheelchair]', *Robotics & Automation Magazine, IEEE*, vol. 8, no. 1, pp. 46-56.

S. McGarry, L. Moir & S. Girdler,(2012), 'The Smart Wheelchair: Is It an Appropriate Mobility Training Tool for Children with Physical Disabilities?', *Disability and rehabilitation. Assistive technology*, vol. 7, no. 5, pp. 372-80.

A. Mihailidis, P. Elinas, J. Boger & J. Hoey,(2007), 'An Intelligent Powered Wheelchair to Enable Mobility of Cognitively Impaired Older Adults: An Anticollision System', *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 15, no. 1, pp. 136-43.

W.T. Miller, R.S.Sutton & a. P.T.Werbos,(1990), 'Neural Networks for Control', *Cambridge, MA:MIT Press 1990*.

J. Minguez & L. Montano,(2000), 'Nearness Diagram Navigation (ND): A New Real-time Collision Avoidance Approach', *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 3, pp. 2094-100 vol.3.

J. Minguez & L. Montano,(2004), 'Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios', *Robotics and Automation, IEEE Transactions on*, vol. 20, no. 1, pp. 45-59.

E. Mjolsness & D. DeCoste,(2001), 'Machine Learning for Science: State of the Art and Future Prospects', *Science*, vol. 293, no. 5537, pp. 2051-5.

L. Montesano, M. Diaz, S. Bhaskar & J. Minguez,(2010), 'Towards an Intelligent Wheelchair System for Users With Cerebral Palsy', *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 18, no. 2, pp. 193-202.

W.B. Mortenson, W.C. Miller, J. Boily, B. Steele, L. Odell, E.M. Crawford & G. Desharnais,(2005), 'Perceptions of Power Mobility Use and Safety within Residential Facilities', *Canadian journal of occupational therapy. Revue canadienne d'ergotherapie*, vol. 72, no. 3, pp. 142-52.

U. Nations,(2013), 'Factsheet on Persons with Disabilities', <http://www.un.org/disabilities/default.asp?id=18>.

M. Nechyba,(1998), 'Learning and Validation of Human Control Strategy', *Ph.D Dissertation, Carnegie Mellon Uni., Pittsburgh, PA, 1998*.

A.V. Nguyen, L.B. Nguyen, S.S. Wang & H.T. Nguyen, (2011), 'Development of a Bayesian Recursive Algorithm to Find Free-Spaces for an Intelligent Wheelchair.', *33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Boston, Massachusetts USA, 2011, pp. 7250-3.

A. V. Nguyen., L.B. Nguyen, S. S. Wang & H. T.Nguyen (2012), 'Development of a Bayesian Neural Networks to Perform Obstacle Avoidance for an Intelligent Wheelchair', *34 rd Annual International Conference of the IEEE EMBS.*, San Diego, California, USA, pp. 1884-7, Aug. 28 2012-Sept. 1

A. V. Nguyen., L. Nguyen, S. S. Wang & H. T.Nguyen (2013a), 'The Advancement of an Obstacle Avoidance Bayesian Neural Networks for an Intelligent Wheelchair', *35 rd Annual International Conference of the IEEE EMBS.*, Osaka, Kansai, JapanOsaka, Kansai, Japan, pp. 3642-5.

A.V. Nguyen, L.B. Nguyen, S.S. Wang & H.T. Nguyen (2013b), 'Shared Control Strategies for Human - Machine Interface in an Intelligent wheelchair', *35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Osaka, Japan, pp. 3638 - 41.

D. Nguyen & B. Widrow,(1990), 'Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights', *IJCNN International Joint Conference on Neural Networks 1990*, pp. 21-6 vol.3.

J.S. Nguyen, S.W. Su & H.T. Nguyen,(2010), 'Spherical Vision Cameras in a Semi-Autonomous Wheelchair System', *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pp. 4064-7.

J.S. Nguyen, N. Tuan Nghia, Y. Tran, S.W. Su, A. Craig & H.T. Nguyen,(2012), 'Real-time performance of a hands-free semi-autonomous wheelchair system using a combination of

stereoscopic and spherical vision', *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pp. 3069-72.

S.T. Nguyen,(2006), ' Advanced bayesian neural network classifiers of head-movement directions for severely disabled people ', *UTS Theses*.

S.T. Nguyen, H.T. Nguyen & P.B. Taylor,(2006), 'Bayesian Neural Network Classification of Head Movement Direction using Various Advanced Optimisation Training Algorithms', *Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on*, pp. 1014-9.

P. Nisbet,(1996), 'Smart' Wheelchairs for Mobility Training', *Technology and Disability*, vol. 5, pp. 49-62.

P. Nisbet,(2002a), 'Assessment and Training of Children for Powered Mobility in the UK', *Technology and Disability*, vol. 14, no. 4, pp. 173-82.

P.D. Nisbet,(2002b), 'Who's intelligent? Wheelchair, driver or both?', *Control Applications, 2002. Proceedings of the 2002 International Conference on*, vol. 2, pp. 760-5.

B.D. Noyes JM,(2007), 'A self-analysis of the NASA-TLX workload measure', *Ergonomics*, vol. 50(4), pp. 514–19.

M. Nuttin, D. Vanhooydonck, E. Demeester & H. Van Brussel,(2002), 'Selection of Suitable Human-Robot Interaction Techniques for Intelligent Wheelchairs', *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on*, pp. 146-51.

S.P. Parikh, V. Grassi, V. Kumar & J. Okamoto,(2007), 'Integrating Human Inputs with Autonomous Behaviors on an Intelligent Wheelchair Platform', *Intelligent Systems, IEEE*, vol. 22, no. 2, pp. 33-41.

S.P. Parikh, R. Rao, J. Sang-Hack, V. Kumar, J.P. Ostrowski & C.J. Taylor,(2003), 'Human Robot Interaction and Usability Studies for a Smart Wheelchair', *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 4, pp. 3206-11 vol.3.

M.R. Petry, A.P. Moreira, R.A.M. Braga & L.P. Reis,(2010), 'Shared Control for Obstacle Avoidance in Intelligent Wheelchairs', *Robotics Automation and Mechatronics (RAM), 2010 IEEE Conference on*, pp. 182-7.

A. Ranganathan & S. Koenig,(2003), 'A Reactive Robot Architecture with Planning on Demand', *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2, pp. 1462-8 vol.2.

A. Robotics,(2011), 'SRF01 Ultrasonic range finders', *Technical Documentation* <http://www.acroname.com/robotics/parts/R322-SRF01.html>.

T. Röfer,(1998), 'Panoramic Image Processing and Route Navigation', *In Ausgezeichnete Informatikdissertationen 1998*, pp. 132–141. B. G. Teubner, Stuttgart, Leipzig. .

W.S.S.S. Sarles,(2002), 'Neural Network FAQ', *periodic posting to the Usenet newsgroup comp.ai.neural-net*, vol. URL: <ftp://ftp.sas.com/pub/neural/FAQ.html>.

H. Seki, S. Kobayashi, Y. Kamiya, M. Hikizu & H. Nomura,(2000), 'Autonomous/Semi-Autonomous Navigation System of a Wheelchair by Active Ultrasonic Beacons', *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 2, pp. 1366-71 vol.2.

R. Simpson, E. LoPresti, S. Hayashi, I. Nourbakhsh & D. Miller,(2004), 'The Smart Wheelchair Component System', *Journal of Rehabilitation Research and Development*, vol. 41, no. 3B, pp. 429-42.

R.C. Simpson,(2005), 'Smart Wheelchairs: A Literature Review', *Journal of Rehabilitation Research and Development*, vol. 42, no. 4, pp. 423-35.

R.C. Simpson & S.P. Levine,(1997), 'Adaptive Shared Control of a Smart Wheelchair Operated by Voice Control', *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 2, pp. 622-6 vol.2.

R.C. Simpson & S.P. Levine,(1999), 'Automatic Adaptation in the NavChair Assistive Wheelchair Navigation System', *IEEE Trans Rehabil Eng*, vol. 7, no. 4, pp. 452-63.

R.C. Simpson, E.F. LoPresti & R.A. Cooper,(2008), 'How Many People Would Benefit from a Smart Wheelchair?', *Journal of Rehabilitation Research and Development*, vol. 45, no. 1, pp. 53-71.

R.C. Simpson, D. Poirot & F. Baxter,(2002), 'The Hephaestus Smart Wheelchair system', *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 10, no. 2, pp. 118-22.

R. Tang, C. Xiao Qi, M. Hayes & I. Palmer,(2012), 'Development of a Navigation System for Semi-Autonomous Operation of Wheelchairs', *Mechatronics and Embedded Systems and Applications (MESA), 2012 IEEE/ASME International Conference on*, pp. 257-62.

M.R.M. Tomari, Y. Kobayashi & Y. Kuno,(2012), 'Development of Smart Wheelchair System for a User with Severe Motor Impairment', *Procedia Engineering*, vol. 41, no. 0, pp. 538-46.

E. Trefler, S.G. Fitzgerald, D.A. Hobson, T. Bursick & R. Joseph,(2004), 'Outcomes of Wheelchair Systems Intervention with Residents of Long-Term Care Facilities', *Assistive Technology*, vol. 16, no. 1, pp. 18-27.

H.T. Trieu, H.T. Nguyen & K. Willey 2007, 'Obstacle Avoidance for Power Wheelchair Using Bayesian Neural Network', *Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Vols 1-16, 2007.*, pp. 4771-4.

H.T. Trieu, H.T. Nguyen & K. Willey 2008a, 'Advanced Obstacle Avoidance for a Laser Based Wheelchair Using Optimised Bayesian Neural Networks', *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Vols 1-8*, pp. 3463-6.

H.T. Trieu, H.T. Nguyen & K. Willey 2008b, 'Shared Control Strategies for Obstacle Avoidance Tasks in an Intelligent Wheelchair', *2008 30th Annual International Conference of the Ieee Engineering in Medicine and Biology Society, Vols 1-8*, pp. 4254-7.

H.T. Trieu, K. Willey & H.T. Nguyen,(2009), 'Adaptive Shared Control Strategies Based in the Bayesian Recursive Technique in an Intelligent Wheelchair', *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pp. 7118-21.

P. Trust,(2011), 'Disability in the United Kingdom 2011: Facts and Figures', *www.papworth.org.uk, July 2011*.

C. Urdiales, M. Fernandez-Carmona, J.M. Peula, U. Cortes, R. Annichiaricco, C. Caltagirone & F. Sandoval,(2011), 'Wheelchair Collaborative Control for Disabled Users Navigating Indoors', *Artificial Intelligence in Medicine*, vol. 52, no. 3, pp. 177-91.

C. Urdiales, B. Fernandez-Espejo, R. Annicchiarico, F. Sandoval & C. Caltagirone,(2010), 'Biometrically Modulated Collaborative Control for an Assistive Wheelchair', *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 18, no. 4, pp. 398-408.

D. Vanhooydonck, E. Demeester, A. Huntemann, J. Philips, G. Vanacker, H. Van Brussel & M. Nuttin,(2010), 'Adaptable Navigational Assistance for Intelligent Wheelchairs by Means of an Implicit Personalized User Model', *Robotics and Autonomous Systems*, vol. 58, no. 8, pp. 963-77.

P. Viswanathan, J.J. Little, A.K. Mackworth, A. Mihailidis & Acm 2011, *Navigation and Obstacle Avoidance Help (NOAH) for Older Adults with Cognitive Impairment: A Pilot Study*.

T.E. Wan LM,(2010), 'Power Wheelchair Assessment and Training for People with Motor Impairment', *12th Intl. Conf. on Mobility and Transport for Elderly*.

C.M. Wang,(1988), 'Location Estimation and Uncertainty Analysis for Mobile Robots', *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pp. 1231-5.

B. Woods & N. Watson,(2003), 'A Short History of Powered Wheelchairs', *Assistive Technology*, vol. 15, no. 2, pp. 164-80.

M. Yokozuka, Y. Suzuki, N. Hashimoto & O. Matsumoto,(2012), 'Robotic Wheelchair with Autonomous Traveling Capability for Transportation Assistance in An Urban Environment', *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 2234-41.

M. Yuwono, A.M.A. Handojoseno & H.T. Nguyen,(2011), 'Optimization of head movement recognition using Augmented Radial Basis Function Neural Network',

Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE, pp. 2776-9.