• U C •

Hugo André Domingues Paiva

Human-Machine Interface Modules to Support Indoor Navigation of a Robotic Wheelchair

Dissertation submitted in partial fulfillment of the requirements for
the degree of Master in Electrical and Computer Engineering
September, 2015

• U C •

UNIVERSIDADE DE COIMBRA

University of Coimbra
Faculty of sciences and technology
Departament of Electrical and Computer Engineering

Human-Machine Interface Modules to Support Indoor Navigation of a Robotic Wheelchair

Hugo André Domingues Paiva

Coimbra, 2015

Human-Machine Interface Modules to Support Indoor Navigation of a Robotic Wheelchair

Advisor: Prof. Dr. Urbano José Carreira Nunes
Co-advisor: Dr. Ana Cristina Barata Pires Lopes

Hugo André Domingues Paiva

Submitted in partial fulfillment of the requirements for the degree of Master in Electrical and Computer Engineering

Department of Electrical and Computer Engineering

Faculty of Sciences and Technology, University of Coimbra

September, 2015

“The starting point of all achievement is desire.”

-Napoleon Hill.

Acknowledgements

Completing a project like this could not have been made without the help of friends, professors and family. I would like to express gratitude to my advisors Professor Dr. Urbano Nunes and Dr. Ana Lopes, for having the patience necessary and always giving me the best advises.

I thank ISR for providing the excellent conditions and resources that allowed me to accomplish my dissertation and also to my colleagues of ISR in special to Daniel Almeida, Jorge Perdigão and David Silva; they were a great help in the hardware assembly.

I would also like to express gratitude to my friends for the moments we shared in the past years.

A special thanks to my girlfriend Marisa for the patience, support and comprehension in my studies.

And finally a big "Thank You" to my parents and my grandmother for providing me with everything that was necessary to finish my studies and of course their support was truly important.

Resumo

Os joysticks convencionais são das interfaces mais utilizadas em cadeiras de rodas eléctricas. No entanto, muitos utilizadores têm dificuldades em manobrar a cadeira usando o joystick na sua forma convencional, que consiste no controlo direto da cadeira de rodas eléctrica. O desenvolvimento de um robô de assistência capaz de proporcionar segurança, e mitigar algumas dificuldades que o utilizador possa possuir, pode constituir uma solução para esse problema. Nesta dissertação propõe-se um robô de assistência, mais especificamente uma cadeira robótica que foi completamente re-projectada com o objectivo de fornecer capacidades a nível de hardware para suportar o novo software proposto e assim proporcionar segurança, mitigar tremores nas mãos e oferecer um modo discreto de condução da cadeira robótica, para utilizadores incapazes de manobrar a cadeira de forma continua com o joystick.

Em termos de hardware, a RobChair Storm foi baseada na RobChair 2.0 que é usada no ISR para fins de investigação. Deste modo, desenvolveu-se um sistema simples, versátil e fácil de usar. O sistema de navegação assistida é baseado num sistema de controlo partilhado que tem em conta as intenções do utilizador e é capaz de proporcionar uma navegação segura em ambientes interiores. O sistema recebe a informação do ambiente envolvente através de um sensor Kinect. Os dados colectados são pre-processados de forma a criar um mapa local. Da intenção do utilizador é determinado um objetivo de navegação. Utiliza-se um planeador global para determinar um caminho global para esse objetivo e um planeador local para lidar com o ambiente dinâmico (e.g. desvio de obstáculos). Todos os algoritmos propostos foram desenvolvidos e testados em ROS.
Esta dissertação foi realizada no âmbito dos projetos "RECI/EEI-AUt/0181/2012-AMS-HMI12: Apoio à Mobilidade Suportada por Controlo Partilhado e Interfaces Homem-Máquina Avançados " e "Centro-07-ST24-FEDER-002028: Diagnosis and Assisted Mobility for People with Special Needs" financiados pela Fundação para a Ciência e Tecnologia (FCT), FEDER, e programas QREN e COMPETE.

Palavras-chave: Joystick, robô de assistência , Robô, Navegação, Cadeira de rodas, Cadeira de rodas robótica, RobChair Storm, ISR, ROS, Mapas locais, Planeador Global, Planeador Local.

Abstract

Conventional joysticks are still the most frequent interface between a human and a Powered Wheelchair (PW), even though many users find it extremely complicated to use the joystick in the standard way, corresponding to the direct control of the PW. The solution to this problem may be the design of an assistive robot capable of providing safety and mitigate some difficulties that the user might have. In this dissertation we describe an assistive robot, more specifically a Robotic Wheelchair (RW) that was completely redesigned with the aim of providing the hardware capability to support the software created to provide safety, mitigate hand tremors and offering a discrete driving of the RW of users incapable of providing a continuous steering command.

In terms of hardware, the RobChair Storm was based in the already assembled RobChair 2.0 that is used in ISR for research purposes. Giving a system that is simple and versatile, making it easy to use and understand. The assistive navigation system (ANS) is based on a shared-control system that takes into account user's intent and is able to provide a safe navigation through the indoor environment. The system receives the information of the surrounding environment through a Kinect. The data collected is preprocessed in order to create the local map with that information. From the user command a sublocal goal is determined and global path is computed with a global planner and the obstacle avoidance problem is surpassed with a local planner. All these algorithms are developed and tested under the framework ROS (Robot Operating System).

This dissertation was developed in the scope of the projects "RECI/EEI-AUt/0181/2012-AMS-HMI12: Support to mobility Supported by a Shared-Control and advanced Human Machine Interfaces" and "Centro-07-ST24-FEDER-002028: Diagnosis and Assisted Mobility for People with Special Needs" funded by Fundação para a Ciência e Tecnologia (FCT), FEDER e programas QREN e COMPETE.

Key words: Joystick, Assistive robot, Robot, Navigation, Wheelchair, Robotic Wheelchair, RobChair Storm, ISR, ROS, Local Mapping, Global Planner, Local Planner.

Contents

Acknowledgements **i**

Resumo **ii**

Abstract **iv**

List of Figures **ix**

List of Tables **xi**

Nomenclature **xiii**

1 Introduction **1**

1.1 Motivation and context 1

1.2 Goals 2

1.3 Implementations and key contributions 3

2 Background and state of the art **5**

2.1 Assistive robotic wheelchairs 5

2.2 HMI in assistive robotics 6

2.2.1 Joystick as HMI 7

2.3 Path planning 8

2.3.1 Global planner 8

2.3.1.1 Dijkstra's graph search algorithm 9

2.3.2 Local planner 9

2.3.2.1 Dynamic window approach 9

2.4 RANSAC algorithm 11

2.4.1 Plane detection in pointcloud data 12

3 Assistive Navigation System **13**

3.1 ANS architecture 13

3.2 Planning system 13

3.2.1 Global planner 13

3.2.2 Local planner 15

3.3 Kinect based perception 15

3.3.1 Pointcloud to fake laser scan 15

3.3.2 Ground subtraction 16

3.3.3 Costmap 17

3.3.4 ROS integration 17

4 RobChair Storm platform **21**
4.1 RobChair Storm . . . 21
4.1.1 System Overview . . . 22
4.1.1.1 Arduino Board: Joystick and Stepper Motor . . . 22
4.1.1.2 Kinect Xbox . . . 24
4.1.1.3 Rotational base . . . 24
4.1.1.4 MDC2230 Motor Controller From Roboteq . . . 26
4.1.1.5 Raspberry Pi . . . 26
4.2 PID Controller . . . 28
4.2.1 1st Order Transfer Function . . . 28
4.2.2 PID Gains . . . 28
4.2.3 PI Validation . . . 31

5 HMI: Joystick Modules **33**
5.1 Manual Mode . . . 33
5.2 Semi-autonomous mode . . . 34
5.2.1 Kinematic model of robotic wheelchairs . . . 35
5.2.2 General User intent . . . 35
5.2.3 User intent with Hand Tremor Filter . . . 36
5.2.4 User intent with discrete driving mode . . . 39

6 Tests and Results **41**
6.1 Kinect based perception system . . . 41
6.1.1 2.5D fake laser scan limitations . . . 41
6.1.2 Kinect Visibility . . . 42
6.1.3 Map update time . . . 42
6.2 ANS tests . . . 43
6.2.1 ANS test in simulated environment . . . 43
6.2.2 Experimental cases . . . 45
6.2.3 Backwards safety . . . 46
6.2.4 Experimental cases discussion . . . 47
6.3 Limitations of the ANS . . . 48

7 Conclusions and future work **49**
7.1 Conclusions . . . 49
7.2 Future work . . . 50

Bibliography **51**

List of Figures

1.1 ANS main modules and key contributions 3

2.1 Trajectory generation for linear and angular velocities 10
2.2 RANSAC: Detection of the points corresponding to the mathematical model of a line in the polar form, with a threshold d. 12
2.3 RANSAC Algorithm: Plane detection 12

3.1 RobChair Storm Assistive Navigation System 14
3.2 Navigation Stack: Global Planner and Local Planner. 15
3.3 Pointcloud to laser scan 16
3.4 RANSAC: Ground Subtraction from pointcloud 16
3.5 Interconnected ROS nodes of the ANS 18
3.6 RobChair Storm transforms tree 18

4.1 RobChair Storm Dimensions 21
4.2 Power connections of the devices presented in RobChair Storm 23
4.3 Arduino Board and connections to Joystick and Stepper Motor 24
4.4 Kinect's Structure 25
4.5 RobChair Storm 25
4.6 Rotational Platform to rotate Kinect 26
4.7 MDC22xx Motor Controller from Roboteq [Rob, 2015] 26
4.8 Raspberry Pi in RobChair Storm 27
4.9 Simulink Diagram for 1st order transfer function of RobChair Storm's Motors 28
4.10 Motor response for a 30% maximum speed input command 29
4.11 Simulink Diagram for simulating the PID Controller applied to first order model of RobChair Storm Motor 29
4.12 Simulated Motor response and real motor response with a PI controller 31

5.1 Joystick Modules taxonomy 33
5.2 Joystick Illustration and his Axis 34
5.3 Blue line is the relationship between the linear velocity and V_y, Red line is the relationship between the angular velocity and V_x 34
5.4 Examples of goals created from the Joystick commands: vertical slider controls linear velocities and horizontal slider controls angular velocities 36
5.5 Kalman Filter with Q = [0.1 0; 0 0.1], the red wave is the user command and blue wave is the user command filtered 37
5.6 Kalman filter with Q = [0.001 0; 0 0.001] 38
5.7 Kalman Filter with Q = [0.0001 0; 0 0.0001] 38
5.8 Rapid movement example, Kalman Filter with Q = [0.0001 0; 0 0.0001] 38
5.9 Diagram of the Discrete Driving Algorithm 39

5.10 Discrete Driving Example: User Command and Cancel Command 40
5.11 Discrete Driving Example: Command Canceled after t>10s 40

6.1 Example of a real scenario for which the 2.5D fake laser may be a problem, due to the occlusion of the table. 41
6.2 User intent of going forward in the scenario presented in Fig. 6.1 created with the laser scan (on the left) and with the pointcloud (on the right) 42
6.3 Scenario where the obstacle is not visible by the Kinect if it has a 0 degree tilt angle. . 42
6.4 Kinect mounted with a tilt angle of -20 degrees. 43
6.5 Navigation task consisting of moving the RW from one room to another. 44
6.6 Linear and angular user commands and real velocity sent to the wheels 44
6.7 Scenario of the tests performed to the ANS. 44
6.8 Path followed in experimental case 1 45
6.9 Path followed in experimental case 2 45
6.10 Path followed in experimental case 3 46
6.11 Path followed in experimental case 4 46
6.12 Backwards safety experimental case 46
6.13 Problem of passing a door way 48

List of Tables

1.1 Most Valued features in a PW 2
1.2 Limiting factors of PW use 2
1.3 Possible improvements 2

2.1 Recent wheelchair platforms. 6
2.2 Joystick as HMI 8

4.1 Components presented in RobChair Storm 22
4.2 1st order model parameters 28

6.1 Pointcloud vs 2.5D fake laser scan 43
6.2 Metrics obtained for the experimental cases 47

Nomenclature

AMR	Assistive Mobile Robotic
ANS	Assistive Navigation System
APCC	Assistive Mobile Robotic
API	Application Programming Interface
CAN	Controller Area Network
DWA	Dynamic Window Approach
HDMI	High Definition Media Interface
HMI	Human-Machine Interaction
ISR	Institute of Systems and Robotics
KBPS	Kinect Based Perception System
OS	Operating System
PID	Proportional-Integral-Derivative
PW	Powered Wheelchair
RANSAC	RANdom SAmple Consensus
RGB	Red, Green, and Blue
RobChair	Robotic Wheelchair (Referring to the ISR Robotic Wheelchair)
ROS	Robot Operating System
RW	Robotic Wheelchair
TCP	Transmission Control Protocol
USB	Universal Serial Bus

Chapter 1

Introduction

This chapter presents an introduction to this dissertation. Some insights concerning the motivation and context of the developed work will be presented, as well as the main goals and key contributions.

1.1 Motivation and context

People with cognitive and physical impairments are sometimes unable to fully control powered wheelchairs, even to navigate the wheelchair to the desired destination, especially under the limited space or surrounding complex structures [Tsu-Zen Hong, 2013]. To this end RWs are being designed and developed with the aim to improve the quality of life of people suffering from motor disabilities. These RWs aim to assist users who are suffering from mobility impairments to achieve a level of autonomy and mobility so that the basic activities of daily living can be retained.

The work presented in this dissertation is framed in the RobChair project , which aims to research and develop technology for robotic wheelchairs. This project is being developed since the mid-1990s with the aim of providing motor impaired users with greater levels of autonomy. Since then, robotic wheelchair hardware and software have been completely restructured three times.
[Pires and Nunes, 2002] used a shared-memory strategy to manage the data provided by the low-level system based on a MC68332 microcontroller and the high-level system.
In [Sousa et al., 2007], [Lopes et al., 2007], [Lopes et al., 2013] the wheelchair included an industrial embedded PC, mounted in the front, powered by Linux with RTAI for real-time processing. The embedded PC was connected to sensors and actuators through CAN fieldbus. The previous architecture of the RobChair had the advantage of being modular, with the CAN bus allowing new devices to be connected to the system. However, this architecture became complex, due to several microcontrollers accessing the CAN bus, which contributed to a higher failure rate of the system, also making the learning curve of a new researcher quite high. In [Gonçalves, 2013] a new and simpler architecture was proposed and developed. Additionally, the robotic Operating System (ROS) was also integrated in this new platform, which allows easily reusing and exchanging code from a broader scientific community.
In order to extend the RW accessibility, several Human-Machine Interfaces (HMIs) have been incorporated over time. In [Pires and Nunes, 2002] the RW is steered through voice commands, allowing its use by severe motor impaired users. A conventional joystick is used in [Sousa et al., 2007], and an altered joystick to allow its use by people with hand tremors is proposed in [Urbano et al., 2009].
[Lopes et al., 2013] proposes a brain-actuated RW based on a synchronous Brain Computer Interface.

The latest RobChair developments were performed with the collaboration of the Portuguese Association of Cerebral Palsy. In fact, patients with CP (Cerebral Palsy) are a paradigmatic example of motor impaired users. CP is the result of a brain injury or a brain malformation, an individual with CP will show signs of physical impairment because CP affects muscles and a person's ability to control them.

In [Ana Carvalho, 2014] a questionnaire for mobility, accessibility and safety characterization was carried out, the questionnaire included 16 individuals and was collected at APCC. With this study was possible to understand what are the most valued features (see Table 1.1), limiting factors (See Table 1.2) and potential improvements required by Powered Wheelchair (PW) users (See Table 1.3).

Most Valued features in a PW	% of users that answered positively
Easy navigation and PW control	56,25%
Safety	43,75%
Dimension	12,50%

Table 1.1: Most Valued features in a PW

Limiting factors of PW use	% of users that answered positively
Difficult in reverse driver	37,50%
Dimension	18,75%
Safety	12,50%
Complicated interfaces	6,25%

Table 1.2: Limiting factors of PW use

When asked if the PW could be improved 93,75% answered positively.

Possible Improvements	% of users that answered positively
Aid for reverse driver	40%
Collision Avoidance	26,67%
Reverse driving information	26,67%

Table 1.3: Possible improvements

This research also shoved that most of the subjects (81,25%) use a joystick as HMI.

1.2 Goals

This work aims to assembly a RW at electric and electronic levels and by maintaining the mechanical structure, and development of a HMI.

The goals for this dissertation were:

1. Create a physical platform that could provide the hardware required to deploy an appropriate solution that meets the valued features and limiting factors pointed out by the PW users [Ana Carvalho, 2014].

2. Implementing in ROS an algorithm to interpret user intent and creating a joystick-based HMI module to mitigate tremor hands and fatigue caused by the continuous use of joystick.

3. Processing data acquired from Kinect and build local costmaps for local navigation.

4. Integrate a ROS package to perform local navigation in a safe way.

5. Perform a set of tests on RobChair Storm to validate the proposed methodology.

1.3 Implementations and key contributions

Figure 1.1 illustrates the main modules developed and described in this dissertation.

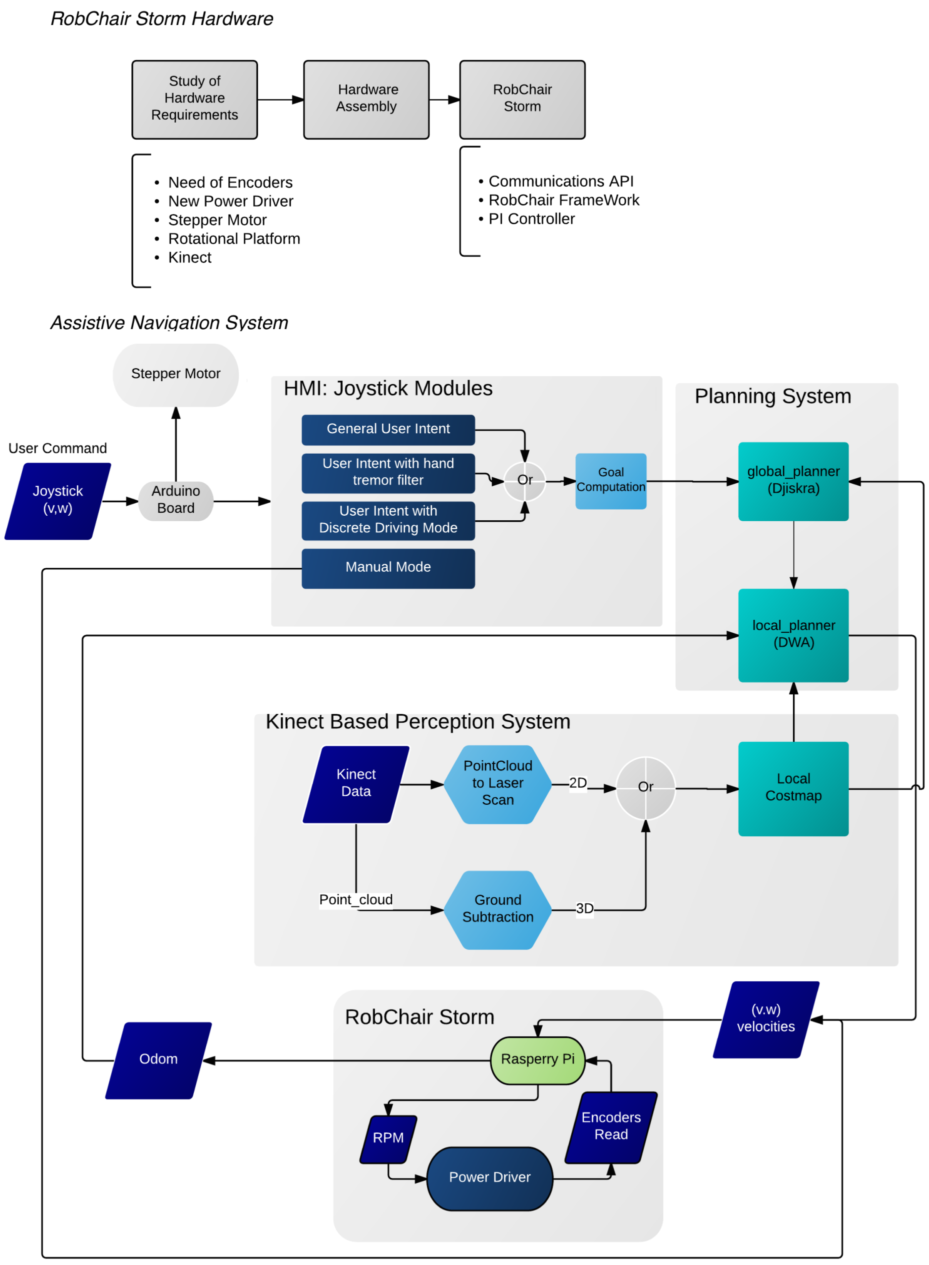


Figure 1.1: ANS main modules and key contributions

Planning System (Chapter 3):

- *move_base* package: Integration of a ROS package, *move_base* package, to perform local navigation.

Kinect-based perception system (Chapter 3):

- Ground Subtraction: Detection of points within a point cloud that support a plane model with the RANSAC method and points elimination.
- 3D Kinect data to 2.5D fake laser scan: A ROS package was integrated to transform 3D scans into 2.5D scans.

RobChair Storm (Chapter 4):

- Hardware assembly: Assembly of the wheelchair electronic hardware, such as Computer, Raspberry pi, Arduino, Kinect, Stepper motor and power driver.
- Communications API: Adaptation of the communication software used on RobChair 2.0 to communicate between low and high level devices presented in RobChair Storm (Computer, Raspberry Pi, Power driver).
- RobChair “Operating System”: A ROS node that is able to manage connections between RobChair Storm devices and allows communication with other external devices such as joysticks, tablets etc.. This node was developed in [Gonçalves, 2013].
- Arduino Board: An Arduino board was used to receive joystick commands and send them to ROS.
- Stepper Motor and Rotational base: A stepper motor was used to rotate the Kinect it was also necessary to design a rotational base to make it possible.
- PI controller design: PI controller design and implementation for RobChair Storm motion controller was carried out.

Human-Machine-Interface: Joystick Modules (Chapter 5):

- General User intent: A ROS node was created to receive and interpret the user intent.
- User intent with hand tremor filter: The Kalman Filter was applied to the joystick commands to mitigate any hand tremor.
- User intent with Discrete Driving mode: A node was created to provide a discrete driving of RobChair Storm.

In Chapter 6, tests and the results to the system are performed and discussed.

Chapter 2

Background and state of the art

In this chapter the concepts related to the work developed in this dissertation will be introduced. The first section presents the concept of assistive mobile robotic (AMR). The following sections are a review of the main subject addressed in this dissertation, namely HMI in assistive robotics and the background theory that support some important issues used to develop the work.

2.1 Assistive robotic wheelchairs

An assistive mobile robot (AMR) can be defined as presented below:

- An assistive mobile robot performs a physical task where its goal is to improve the well-being of a person with a disability. The task is related to normal human activities of a daily living, as for example reach indoor divisions in security. The person that benefits from the task of the assistive robot is the one who controls the functioning of the robot [Jaffe et al., 2012].

- An assistive mobile robot can also be seen as a device that can sense, process sensory information, and perform actions that benefit people with disabilities and Elderly people. An assistive robot is a Human-Centered-Robot System that obeys to a type of Semi-Autonomous control scheme, where both human and machine agents are able to influence the control of the system [Lopes, 2012].

An AMR can improve the quality of life of motor impaired people. For these users the main need that requires assistance is moving the person to a desired location [Miller, 2006]. Minor impairments can often be overcome through the use of canes and crutches while more greater difficulties are usually solved by a manual wheelchair or PW. Although there are still users who can't safely drive a PW. For them an assistance like a RW may prove very beneficial. Robotic Wheelchairs are inserted on a strong field of research and development with major results achieved in the recent years due to the development of microelectronics [Grasse et al., 2010]. These RWs aim to assist users who are suffering from mobility impairments in the most various ways and with different HMI. In extreme cases of motor impairment, a joystick is not a valid HMI. Some projects rely in different types of HMI: [Yanco, 1998] purposed a RW with a graphical user interface that could be controlled through an eye tracking or with a single switch scanning device. Brain-Computer-Interfaces have also been used in the RW field as for example [Lopes et al., 2013], [Perdigão, 2014] and[Bonarini et al., 2012]. A list of some recent Robotic Wheelchairs are presented in Table 2.1, with a description of the main technologies applied, the implemented shared-control type and a joystick as HMI. [Bonarini et al., 2012] also used a Joystick that allows a conventional control of the RW while superimposing ad hoc maneuvers to avoid unforeseen objects. [Patel et al., 2012] used a joystick to learn a mixed array of the Activities of Daily Living (ADL) and provide assistance and when required. [Park et al., 2012] used a joystick to continuously estimate the user's intention and it determines whether the user needs assistance to achieve that intention.

Table 2.1: Recent wheelchair platforms.

Institution	Main Robotic Technologies	Shared-Control Type and user intention	HMI
University of technology of Sydney [Patel et al., 2012]	Montecarlo localization Topological mapping.	Hierarchical Hidden Markov Model framework that predicts both the short term (local) and long term (navigational) goals of the user.	Joystick.
VAHM (LASC, University Paul Verlaine-Metz) [Grasse et al., 2010]	Particle filtering approach to implement the recognition of the most frequent paths according to an offline-constructed topological map.	Provides assistance to the user during navigation by proposing the direction to be taken when a path has been recognized.	Joystick.
SHARIOTO (Katholieke Universiteit Leuven) [Vanhooydonck et al., 2010]	Dynamic window approach for obstacle avoidance.	Shared-control with user intention prediction based on a Bayesian network.	Joystick.
LURCH Politecnico di Milano [Bonarini et al., 2012]	Localization based on odometry. Odometry correction is performed based in the detection of passive markers placed in the environment using vision. Trajectory planning based on the fast planner SPIKE (Spike plans In Known Environments).	Control module based on a fuzzy behavior management system, where a set of reactive behaviors, which will be carried out by the robot, are implemented as a set of fuzzy rules. Two set of rules were established: one implementing trajectory following, and the another one implementing obstacle avoidance.	Joystick, touch-screen, electro miographic interface, and Brain Computer Interface (BCI).
University of Michigan [Park et al., 2012]	A static occupancy grid map obtained via SLAM. Global topological map. Position and velocity estimation of new obstacles in the environment based on a Kalman filter.	Model Predictive Equilibrium Point Control (MPEPC) framework, which allows the navigation of a wheelchair in dynamic, uncertain, structured scenarios with multiple pedestrians.	Joystick.

2.2 HMI in assistive robotics

The Design of HMI for people with cognitive and physical impairments have some major requirements according to [Hillman and Jepson, 1992]. Those requirements can be summarized as follow:

- **The need for simple and intuitive interface:** The communication between the robot and the human must be clear. This means that the HMI must be able to indicate the user intent to the robot using intuitive and natural verbal and gesture tools, the robot must also be able to retrieve is state to the user.
- **The need for perception modeling:** Sensing and predicting the intention of the user is a major step when designing natural human-machine interfaces.
- **Flexible and genuine interface accessibility:** The ability of the operator should also be considered when designing HMI to an assistive robot. For example a person with cognitive and physical impairments has some struggle when doing some physical movements. In these cases designing interfaces that are generic and accessible for all is a considerable challenge.
- **The need for an adequate feedback:** Offering adequate feedback for user's actions has been recognized as a fundamental principle in the design of a good HMI.

- **Keep the user in the control loop of the system:** HMI dedicated to people with cognitive and physical impairments should be intelligent enough to sense the intention of the user and provide a semi-autonomous system instead of an autonomously one. The user should feel that he has the control of the system.
- **Cost:** Of course the design of HMI sophisticated tools and technologies can end up very expensive. A proper compromise between cost and operability of the system should be taken in consideration.

According to [Baklouti et al., 2010] HMI can be classified into two different classes.

The first class of HMI is represented by the modification of the Hardware such as keyboard, joysticks and trackballs. These devices were changed to fit the person's disability. However, disabled people may not conform to a standard pattern, therefore it cannot be a single pattern which fits everyone and considering individual solutions is extremely expensive in practice. Therefore, having interfaces that are intelligent enough to react differently and appropriately to different stimuli are really important when referring to assistive devices.
The second class represents those that doesn't require any hardware change being that their big advantage compared to the ones of the first class. The whole intelligence and adaptability module consists of processing the signals to extract the control command. Thus, with the use of intelligent algorithms and processing tools the essential requirement for matching the system's and user's need can be fulfilled without changing the hardware and without big costs.

In this dissertation the HMI is a second class one. We propose to use a joystick where no modification has been made and the extraction of the user intent is carried out through the use of algorithms responsible for processing the user's commands.

2.2.1 Joystick as HMI

The joystick is the primary interface between a person and the PW. The conventional joystick with the manually operated stick on a dock is commonly used and in some cases it can be enough to provide some users a reliable control of the PW. For other users different forms of joystick might be more suitable, depending on the user's motor and cognitive capabilities. Some methods rely on different motor capabilities of the users (rather than the hands) for example movements of the head, chin or even for the tongue to operate the joystick. Those approaches rely on the same principle. For instances for the tongue Joystick the movements of the tongue are measured by receiving signals from sensors around the mouth which means that like an ordinary joystick a change of the angle gives the user intention [Tsalicoglou and Perrin, 2009].

As referred before for some people the joystick with no hardware or software change is extremely difficult to operate in a safe manner. Two alternatives might be considered to overcome this problem. The joystick remains the same and its output signal is preconditioned before entering the controller [Aylor et al., 1979] or the joystick is changed to a more appropriate one [Fattouh et al., 2004].

In [Fattouh et al., 2004] a force feedback joystick is proposed. The user has complete control of the RW but the control logic makes the force feedback joystick less or more difficult to go in some direction, depending on the distance between the RW and obstacles (the closest to the obstacles the greater is the feedback force).

In [Aylor et al., 1979] a control system that time average hand tremors and is unresponsive to rapid and erratic movements is presented. The signal is amplified, rectified and then time average to reduce the effect of tremor on control. [Tsu-Zen Hong, 2013] intercept joystick commands and modifies them through algorithms, if necessary, and sends them to the motor drivers. In [Urbano et al., 2009] a Legacy Adapted Operation Mode (LAM) is purposed to give a non-continuos driving of a PW where the user gives the start and stop of the PW. [Poorten et al., 2012] purposed the used of conventional joystick and through kinematically correct environmental haptic feedback, two navigation modes are purposed which are a obstacle avoidance mode and collision avoidance mode.

Table 2.2: Joystick as HMI

Institution	Description	Joystick Class
University of Metz [Fattouh et al., 2004]	Force feedback joystick	1st
University of Virginia [Aylor et al., 1979]	Time average hand tremor filter	2nd
Fu Jen Catholic University [Tsu-Zen Hong, 2013]	Intercept joystick commands and validates them	2nd
University of Coimbra [Urbano et al., 2009]	Provides a LAM operation for non-continuos PW driving	2nd
Mechanical Engineering Dept. of K.U. Leuven [Poorten et al., 2012]	Assistance through kinematically correct environmental haptic feedback	2nd

2.3 Path planning

For a mobile robot, the path-planning problem may be defined as the computation of a valid free collision-path, given a map, an initial and final position. A common strategy to deal with this complex planning problem is separated into a global planning and a local planning.

2.3.1 Global planner

The global planner is responsible for creating a high-level plan for the robot to follow and with the objective to reach the goal location. For that it needs the obstacle and cost information contained in the costmap more specifically: information from the robot's localization system and a goal in the map. It is also important to refer that the global planner may not take into account the dynamics or the kinematics of the robot. The goal to be reached may be provided by a HMI.

Some well known global planning solutions compute an optimal shortest path, usually this solutions are based on a graph and tree search algorithms such as A* [Kala et al., 2010, Hart et al., 1968] and Dijkstra's [Dijstra, 1959, Gerkey and Konolige, 2008]. The Dijkstra's Algorithm was the one used in this dissertation.

2.3.1.1 Dijkstra's graph search algorithm

Dijkstra's Algorithm is a graph search algorithm that solves the shortest path problem, it can be used in the most various applications that requires the search for the shortest path such as: Maps, Urban Traffic planning, Routing of telecommunications messages, Robot Navigation etc. The algorithm is as follows:

Algoritmo 2.1 Dijkstra pseudocode

```
// Let $n_1$ being the initial node and Q being the unvisited set of nodes
function Dijkstra(Graph, $n_1$ ):
 for each node $n$ in Graph:
     dist[n] := infinity // initial distance from $n_1$ to node n is set to infinity
     previous[n] :=unvisited // set the other nodes as unvisited
 dist[$n_1$ ] := 0 // initial node as cost 0
 while Q is not empty:
     u = node in Q with smallest dist[]
     remove u from Q
     for each neighbor n of u: // compare the distance of the current node with all the other nodes and retrieve the smallest distance, where n has not been removed from the unvisited set Q
         alt:= dist[u] + dist_between[u,n]
         if alt < dist[n]
          dist[n]:= alt
          previous[n] = u
 return previous[]
```

2.3.2 Local planner

The local planner is responsible for generating velocity commands for the mobile base that will safely move the robot towards the goal. The local planner will try to follow the Global plan while possibly taking in consideration the kinematics and dynamics of the robot as well as the obstacle information stored in the costmap. In order to generate safe velocity commands the local planner makes use of techniques such as the well known method Dynamic Window Approach proposed in [Fox et al., 1997].

2.3.2.1 Dynamic window approach

The DWA method [Fox et al., 1997] discretely samples the velocity space (v, w) of the robot, where v is the linear velocity and w is the angular velocity, to create a set of feasible trajectories as illustrated in Fig 2.1. The green trajectories are the ones that will navigate the robot safely, and the red ones are trajectories that will generate a collision. Obstacles in the closer surroundings of the robot impose restrictions on the rotational and translational velocities, then the velocity is considered admissible if the robot is able to stop before it reaches this obstacle. Letting $dist(v, w)$ being the distance to the closest obstacle and $\dot{v}_b$ and $\dot{w}_b$ being the accelerations for breakage the set V_a of admissible velocities is defined as:

$$V_a = \left\{(v, w)|v \leq \sqrt{2 \cdot dist(v, w) \cdot \dot{v}_b} \wedge w \leq \sqrt{2 \cdot dist(v, w) \cdot \dot{w}_b}\right\} \tag{2.1}$$

The limited accelerations imposed by the motors must also be taken in consideration when choosing the (v, w) velocities. Let t be the time interval during which the accelerations $\dot{v}$ and $\dot{w}$ will be applied and let (v_a, w_a) be the actual velocity. The dynamic windows V_d is defined as (see equation 2.2):

$$V_d = \{(v, w) | v \varepsilon [v_a - \dot{v} \cdot t, v_a + \dot{v} \cdot t] \wedge w \varepsilon [w_a - \dot{w} \cdot t, w_a + \dot{w} \cdot t]\} \tag{2.2}$$

Because the dynamic window is centered around the current velocity all the curvatures outside the dynamic window cannot be reached within the next time interval and thus are not considered for obstacle avoidance.

Let V_s be the space of possible velocities, then the resulting workspace V_r is defined as the intersection:

$$V_r = V_s \cap V_a \cap V_d \tag{2.3}$$

With the resulting workspace V_r determined a velocity is selected from V_r in order to incorporate the criteria **target heading**, **clearance** and **velocity**, the maximum of the objective function (see function 2.4) is computed over V_r.

$$G(v, w) = \sigma(\alpha \cdot heading(v, w)) + \beta \cdot dist(v, w) + \gamma \cdot velocity(v, w)) \tag{2.4}$$

- **Target heading**: The target heading $heading(v, w)$ measures the alignment of the robot with the target direction in other words it favors controls approaching goal;
- **Clearance**: The function $dist(v, w)$ represents the distance to the closest obstacle that intersects with the curvature, which means it favors controls moving far from obstacles;
- **Velocity**: The function $velocity(v, w)$ is used to evaluate the progress of the robot on the corresponding trajectory;

This last three parameters of G, target heading, clearance and velocity are the ones that will influence the behavior of the robot. By maximizing only the parameter heading and the velocity the robot will favor speeds that lead into free space, but it will not make any effort to reach the goal. By maximizing only the parameter target heading, the robot will stop by the first obstacle that blocks its way. Which means that combining all three parameters will make the robot avoid obstacles as fast as it can while advancing towards the goal.

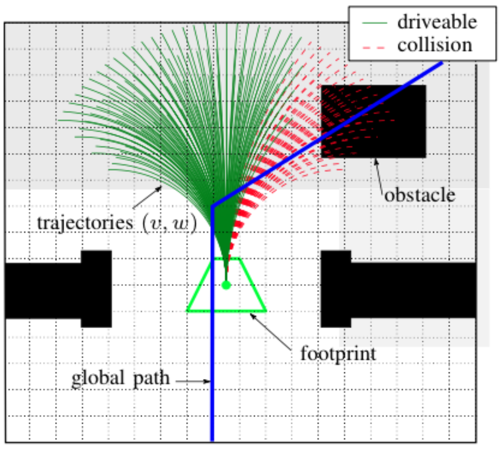


Figure 2.1: Trajectory generation for linear and angular velocities

2.4 RANSAC algorithm

The Random Sample Consensus (RANSAC) is an algorithm proposed by [Fischler and Bolles, 1981] that is used to estimate parameters of a mathematical model from a set of data containing outliers. This algorithm assumes that all data is comprised of both inliers and outliers. Inliers can be explained by a model with a particular set of parameter values, while outliers do not fit that model. It is also necessary to assume that a procedure which can optimally estimate the parameters of the chosen model from the data is available [RAN, 2015].
The input of the RANSAC algorithm is a set of observed data values, a parameterized model which can explain or be fitted to the observations, and some confidence parameters. RANSAC achieves its goal by iteratively selecting a random subset of the original data. This data are hypothetical inliers and this hypothesis is then tested as follows [Wik, 2013]:

1. *A model is fitted to the hypothetical inliers, i.e. all free parameters of the model are reconstructed from the inliers.*
2. *All other data are then tested against the fitted model and, if a point fits well to the estimated model, also considered as a hypothetical inlier.*
3. *The estimated model is reasonably good if sufficiently many points have been classified as hypothetical inliers.*
4. *The model is reestimated from all hypothetical inliers, because it has only been estimated from the initial set of hypothetical inliers.*
5. *Finally, the model is evaluated by estimating the error of the inliers relative to the model.*

This is repeated a fixed number of times, in case that only a few points are classified as inliers the model is rejected, otherwise a refined model together with a corresponding error measured is kept if the error is lower than the last saved model.

In Fig 2.2 an application of the RANSAC algorithm on a 2D set of data is shown. The outliers and the inliers that correspond to the model can be observed, where d is the threshold to be considered inliers or outliers.

In this dissertation the RANSAC Algorithm is used to estimate plane models for ground subtraction in the pointcloud acquired using the Kinect sensor.

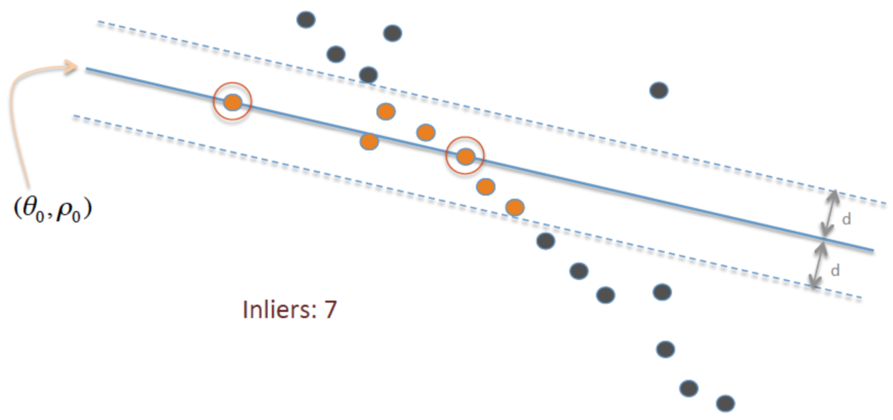


Figure 2.2: RANSAC: Detection of the points corresponding to the mathematical model of a line in the polar form, with a threshold d.

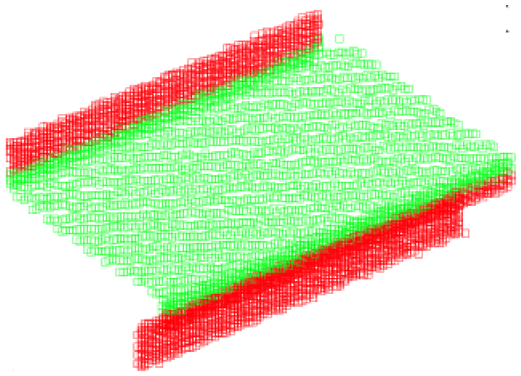


Figure 2.3: RANSAC Algorithm: Plane detection

2.4.1 Plane detection in pointcloud data

The RANSAC algorithm can be implemented to achieve different tasks [Yand and Forstner, 2010]. In [Brenner et al., 2001] the RANSAC algorithm is used to detect the buildings roof planes.
[Schnabel et al., 2007] use RANSAC to detect basic shapes such as planes spheres, cylinders, cones in point clouds. For detecting plane models in 3D point cloud the process is pretty similar as in 2D, in 3D it selects randomly three points and it calculates the parameters (a, b, c and d see 2.5 for the mathematical model of a plane) of the corresponding plane.

$$ax + by + cz + d = 0 \tag{2.5}$$

Then it detects all points of the original cloud belonging to the calculated plane. To confirm the plausibility of the calculated plane the deviation of the plane's normal is determined and the candidate plane is accepted only if all deviations are less than a predefined angle. The Fig. 2.3 shows the RANSAC Algorithm detecting a plane model, where the inliers are represented as green dots and the outliers as red dots.

Chapter 3

Assistive Navigation System

This chapter introduces the Assistive Navigation System (ANS) implemented in RobChair Storm and describes its main modules namely: HMI, Planning and Perception.

3.1 ANS architecture

The ANS system is structured in three main modules, as shown in Fig. 3.1.

- HMI Joystick: It is through these modules, that the user intent is provided. The inputs are the joystick commands in voltage that are interpreted as a linear (v) and angular (w) velocity. The output is a goal calculated from v and w to sent to the global planner.
- Planing: is responsible for determining a global path, with the global planner, to be followed by the local planner while avoiding obstacles. The global planner performs a tree search to find the path, using Dijkstra method. Its inputs are a goal provided by the joystick modules and the global costmap, and the outputs the global path. The local planner implements a dynamic window approach to collision avoidance, having the global path as reference. The inputs are the odometry information and the output is the final linear and angular velocity to be sent to the motors.
- Kinect Based Perception System (KBPS): The perception of the world is made with a Kinect mounted in RobChair Storm. It outputs a fake laser scan or a pointcloud to create the occupancy grid maps, here designated as costmaps.

3.2 Planning system

In terms of conceptual level the planning module considers the data provided from odometry and sensor streams, and outputs velocity commands to be sent to the robot. This planning approach is more suitable to be used in square or circular shape robots, it will perform at reasonable level with other types as for example rectangular shapes, such as RobChair Storm. There are some drawbacks associated to its use that will be discussed in the experimental results sessions. It is this system that will be responsible to provide a safe navigation.

3.2.1 Global planner

When a user sends a command with the joystick, that latter is transformed to a goal. The global planner is now responsible to calculate the shortest path to reach that goal with Dijkstra method, every time a new command arrives a new path is calculated. This method does not takes into consideration the dynamics of the model.

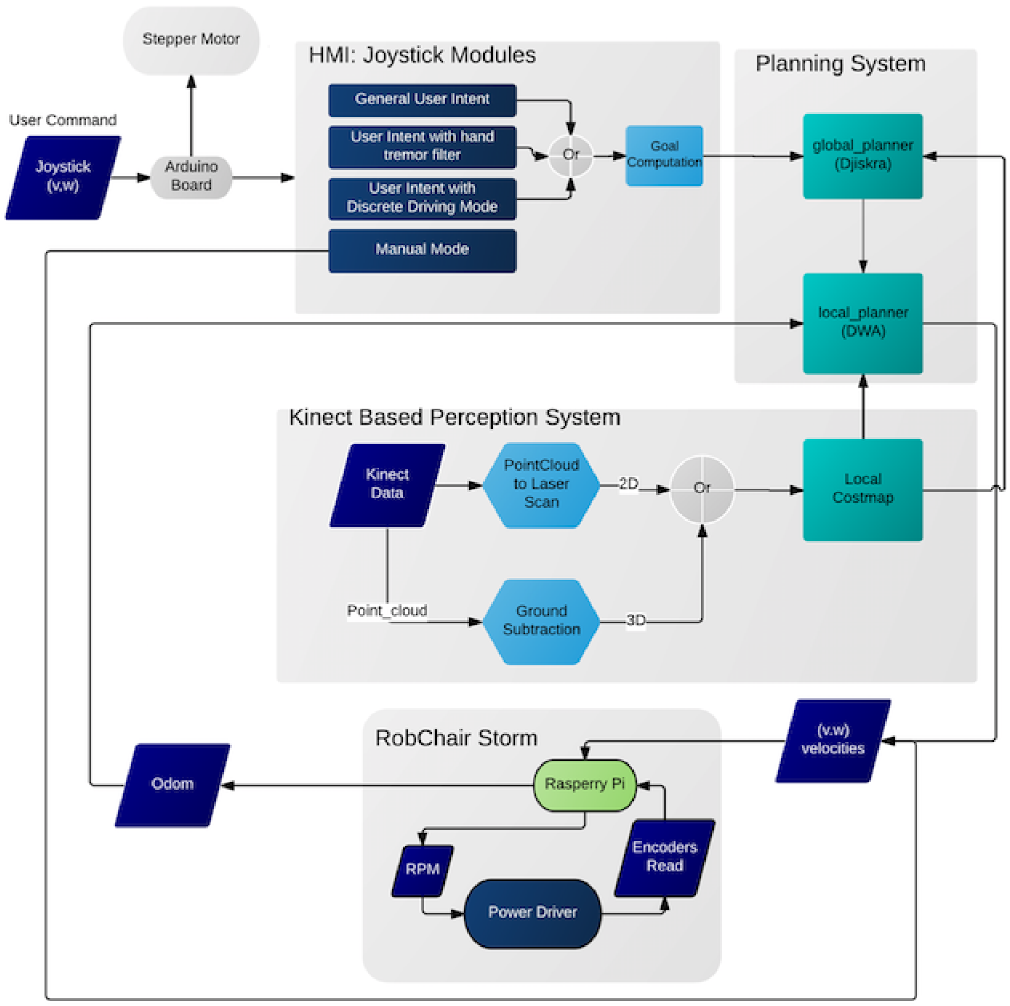


Figure 3.1: RobChair Storm Assistive Navigation System

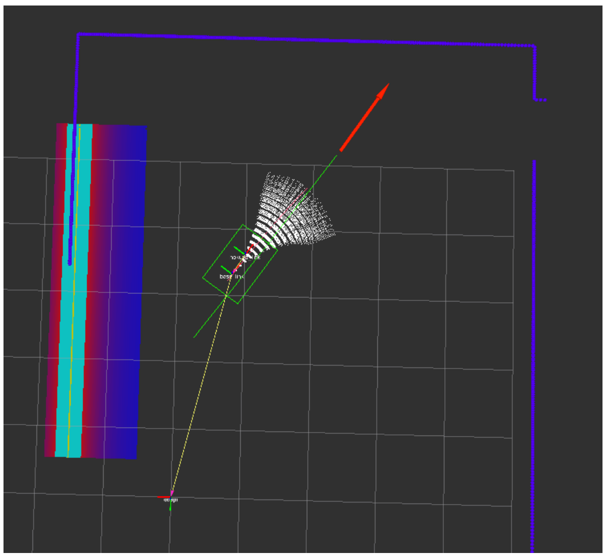


Figure 3.2: Navigation Stack: Global Planner and Local Planner.

3.2.2 Local planner

The Dynamic Window Approach (DWA) algorithm is used to perform local planning. The latter considers the dynamics of the robot and also deals with the constraints imposed by limited velocities and accelerations of the robot. More generally its purpose is to avoid obstacles and follow the global path. Initially it generates the dynamic window which consists of simulating various sets of linear and angular velocities and retrieving the one that avoids the obstacles and follows more closely the global path.

Figure 3.2 shows an example of the navigation system in action, the user sent a command (red narrow), the global planner calculates the global path (green line) and the local planner follows it with the best match (red line) from the dynamic window (white cloud of dots).

3.3 Kinect based perception

The KBPS can provide data in 2.5D or in 3D. It is with this data that the local costmap will be created.

3.3.1 Pointcloud to fake laser scan

From the 3D depth information provided by the Kinect, a fake laser scan is created, which is a projection of the pointcloud to a 2.5D fake laser scan. Basically it takes the closest point of each column of the 3D depth data pointcloud and creates the fake laser scan. Figure 3.3 shows a laser scan created from the 3D depth data.

This approach is relatively good because it condenses a lot of information from the pointcloud to a small laser scan, this will decrease the time to create the occupancy grid and the computational efforts are less. The problem is that sometimes some important information can be discarded.

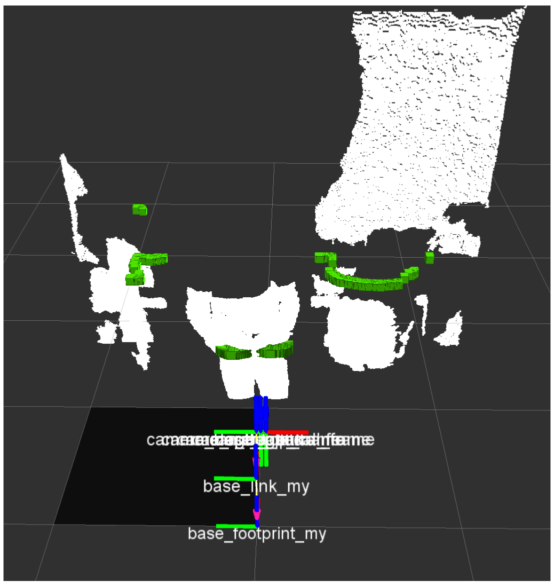


Figure 3.3: Pointcloud to laser scan

3.3.2 Ground subtraction

Other approach to create local costmaps is to use the 3D depth information. With a node that implements the RANSAC algorithm a ground subtraction to the pointcloud is made. To proceed the ground subtraction from a pointcloud it was used the Point Cloud Library (PCL) [PCL, 2015].

Algorithm 3.1 represents in the form of pseudocode the principal classes and functions of PCL library used to perform ground subtraction. This code was developed in ROS and corresponds to the ROS node *Ground_Subtraction*.

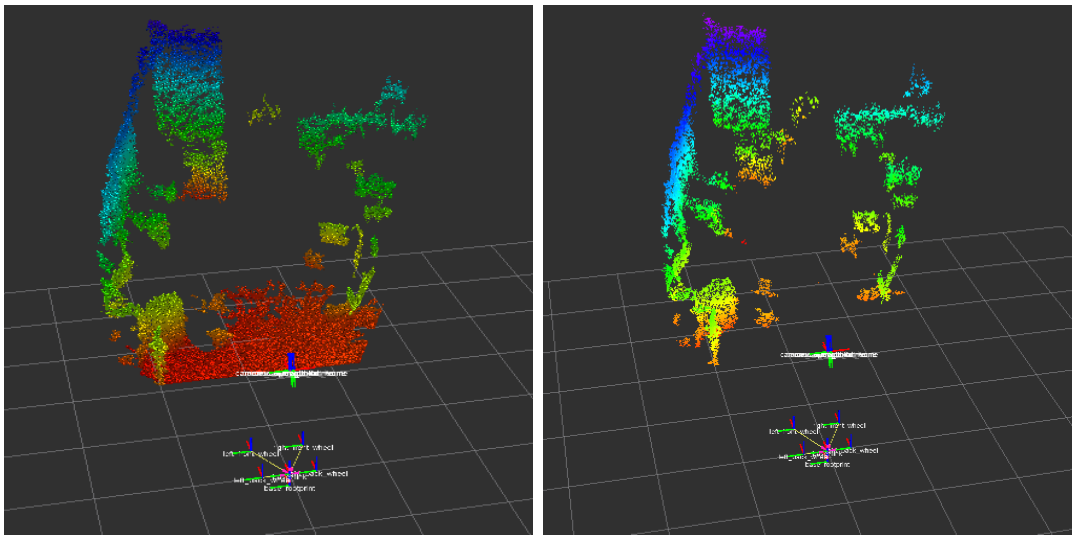


Figure 3.4: RANSAC: Ground Subtraction from pointcloud

Figure 3.4 shows the removing of a plane model from a pointcloud, the left image is the original pointcloud, where it is possible to see the ground plane (red dots). The left image on Fig. 3.4 is the pointcloud that will be used to create the occupancy grid. This pointcloud is the output of the ROS node that implements the RANSAC algorithm. The latter had success on detecting and deleting the

Algoritmo 3.1 Groud Subtraction Algorithm

```
//receives pointcloud from Kinect sensor
loop:
cloud_subscriver(inputCloud)
getPlaneCoefficients(inputCloud)
    SACsegmentation seg // create segmentation object
    seg.setModelType(pcl::SACMODEL_PLANE) // set the model to plane
    seg.setMethodType(pcl::SAC_RANSAC) // Use RANSAC method
    seg.setDistanceThreshold(0.015) // specify threshold which determines how close a point must be to the model in order to be considered an inlier
    seg.setEpsAngle(0.26) maximum allowed difference between the plane normal and the given axis, in radians
    seg.segment() // returns inliers and the model coefficients a,b,c and d
        return coefficients and inliers
removeGroundPlane(inliers, coefficients) // function to remove the points corresponding to the ground plane
    pcl::ExtractIndices extract
    extract.setInputCloud(inputcloud)
    extract.setIndices(inliers)
    extract.filter
        return pointcloud_with_Ground_subtracted
End loop
end procedure
```

ground plane, has it can be observed on the right image of Fig. 3.4. This approach requires more computational efforts to create maps, this will lead to a slower map update compared to the use of fake laser scans. For a system of this type taking to much time updating the map may jeopardize safety.

3.3.3 Costmap

The global planner and local planner require a costmap to be implemented. A costmap maintains information about the free space to where the robot can navigate in the form of an occupancy grid. It uses sensor data to store and update information about obstacles into the environment. Which means that queries about obstacles can only be made in columns. For example if two obstacles share the same position in the XY plane but with different Z positions would result in the same corresponding cell in the costmap. Hence the costmap considers sensor data (laser scans or pointcloud) and builds a 2D or 3D occupancy grid, and inflates costs in a 2D costmap based on the occupancy grid and a user specified inflation radius.

3.3.4 ROS integration

All navigation system is running under ROS. The package *move_ base* was integrated [Marder-Eppstein, 2015] to perform the navigation based on a global planner, local planner and the costmap. A package *depthimage_ to_ laserscan* [dep, 2015] was also integrated to pre-process the 3D data acquired with the Kinect sensor.

Figure 3.5 shows the flux of information between all the nodes that are running in the ANS. All the nodes created and integrated are presented. Note that the node depthimage_to_laserscan and

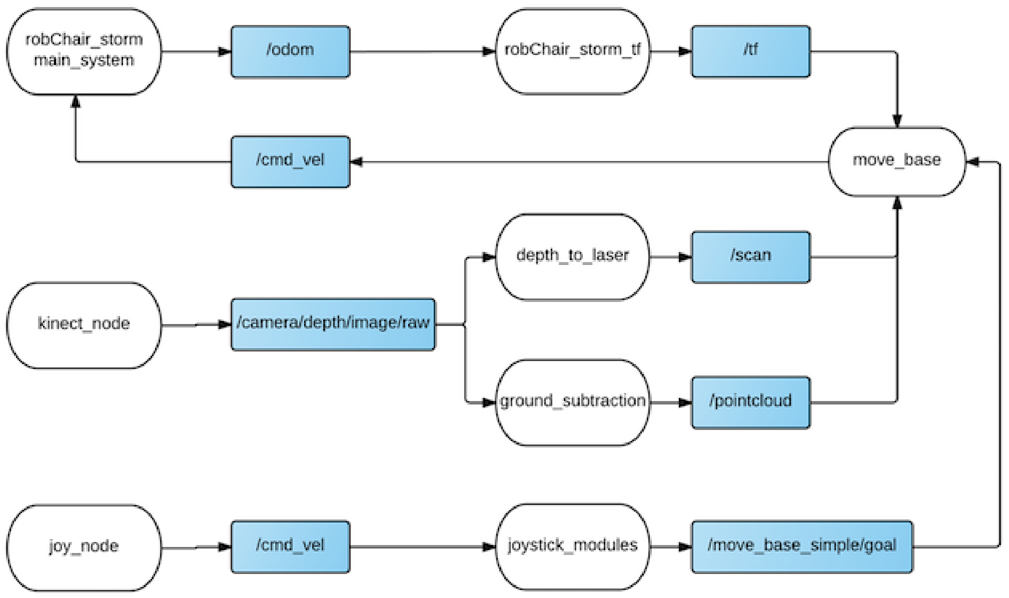


Figure 3.5: Interconnected ROS nodes of the ANS

Ground_Subtraction node never run in parallel and rather are used alternatively.

- *RobChair_Storm_System*: this node is responsible for manage the topics /cmd_vel and /odom. /cmd_vel is the topic where the velocities to send to the wheels are published.
  The *RobChair_Storm_System* node makes the interface with the low level device trough a TCP/IP connection.

- *RobChair_Storm_tf*: it manages all the transforms between frames of RobChair Storm required by the *move_base* node. The Fig. 3.6 shows the transforms tree of RobChair Storm only the two highlighted ones are really important for the system, the rest are only for visualization purposes.

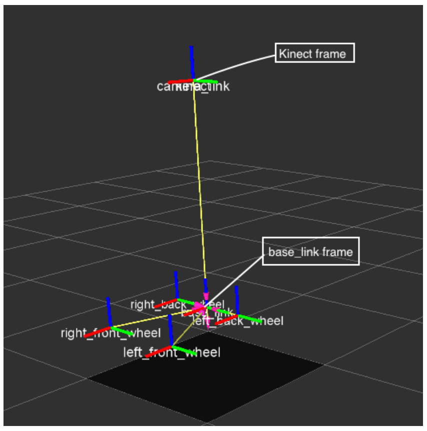


Figure 3.6: RobChair Storm transforms tree

- *HMI_Joystick_Modules*: all the commands from the joystick are received by this node. Responsible for the goal computation which is then sent to the *move_base module* (planning module).

- *Kinect_node*: publishes the Kinect data in the form of point_cloud.

- *depthimage_to_laserscan:* from the pointcloud outputted by the *kinect_node* creates the fake laser scan.
- *Ground_Subtraction:* from the pointcloud published in the topic /camera/depth/image/raw the plane model of the ground is detected and removed.
- *move_base:* this node is responsible for publishing the correct velocities in order to avoid obstacles, with the global planner and local planner as explained before.

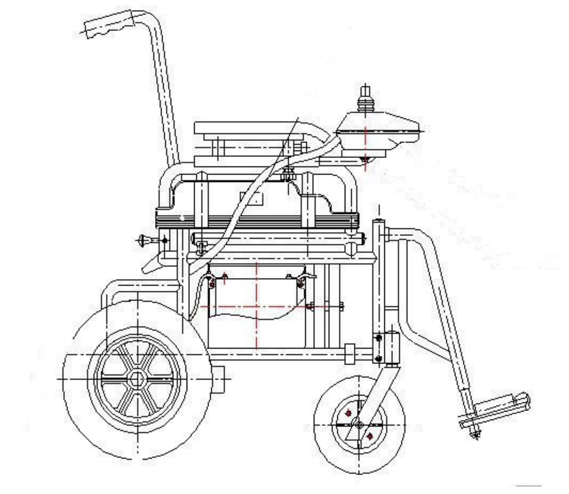
Chapter 4

RobChair Storm platform

This chapter describes all the changes made to the original wheelchair in order to obtain a platform capable of accomplish the objectives proposed for this work. Some of those changes were based in a previous work developed in Robchair [Gonçalves, 2013].

4.1 RobChair Storm

The model provided by APPC is a wheelchair made by Invacare, designed to be mostly used in indoor environments. In the mechanical point of view, the structure is composed by two motorized wheels in the back and two caster wheels in the front. The corresponding dimensions of the wheels, the overall platform and some extra information can be seen in Fig. 4.1.



High	163.2cm
Width	64cm
Depth	84cm
Motorized wheels radius	17.2cm
Maximum motor speeds	1000 RPM
Gear Box reduction factor	1:29
Encoder Pulses per revolution	1000
Distance between Wheels	58cm

Figure 4.1: RobChair Storm Dimensions

The original system was based on an open loop control system, which means that there was not any feedback from the wheel's speed, position or torque, the system was composed by a joystick connected directly to the power driver and the communication was constituted by a communication module based on field-buses, namely Controller Area Network (CAN). This architecture had to be replaced in order to be possible to validate the Joystick commands and in that way to create a safety system. See table 4.1 for the components that make part of the new architecture, a closed loop system.

Component	Quantity	Description
Battery	2	Two 12V batteries connected in series to output 24V and power up the whole system
DC motor	2	Two permanent magnet 24V DC motors
Encoders	2	Two encoders with 1000ppr
Arduino Board	1	One Arduino Board
H-Bridge	1	One L928 Dual H-Bridge
Stepper Motor	1	One Bipolar stepper motor with 200ppr (1.8º of resolution)
Power driver	1	One Roboteq MDC2230 power driver
DC-DC converter	2	One of 24-12V to power the Kinect and the other 24-5V to power the stepper motor
Xbox 360 Kinect	1	One Xbox 360 Kinect
Raspberry Pi	1	One Raspberry Pi, Model B (ARM architecture)
Joystick	1	One analog joystick
Rotational base	1	One rotational base to rotate the Kinect

Table 4.1: Components presented in RobChair Storm

Almost every electric component is powered by the two batteries connected in series except for the Raspberry Pi and Arduino Uno, both these components are powered through a USB connection to the Computer. The Encoders are powered with 5V by the MDC2230 controller, and are connected to the specific encoder channel ports. Figure 4.2 shows all the electrical connections presented in RobChair Storm.

4.1.1 System Overview

The following subsections give some details of the system components.

4.1.1.1 Arduino Board: Joystick and Stepper Motor

The Arduino is a board which has an Atmel AVR micro-controller with complementary components that provide sets of digital and analog I/O pins that can be interfaced to various extensions boards and other circuits. In RobChair Storm its purpose is to receive information from Joystick and to control the Stepper Motor using a L298 Dual H-bridge.

The joystick has two potentiometers powered with 5V. When the user interacts with joystick a change in the Voltage occurs, that change will be received in the Arduino board by the analog pins, because the Arduino board has 10-bit analog to digital converter, the conversion to a 0-5V is made with the next equation:

$$V_{x,y} = \frac{InputVoltage * 5}{1023}$$

This values will then be sent to the Computer where in function of V_y a linear velocity is calculated and in function of V_x an angular velocity is calculated.

The Stepper Motor is used to rotate the Kinect to a certain position where the referred position is relative to the user command, for example if the user chooses a negative linear velocity the Stepper rotates the Kinect to the position 180º giving a secure navigation to the platform in backwards. Like it has been referred before the control of the Stepper can't be done by the Arduino board alone in this thesis was used the L298 which is a dual H-bridge driver designed to drive inductive loads such as relays, solenoids, DC and stepping motors.

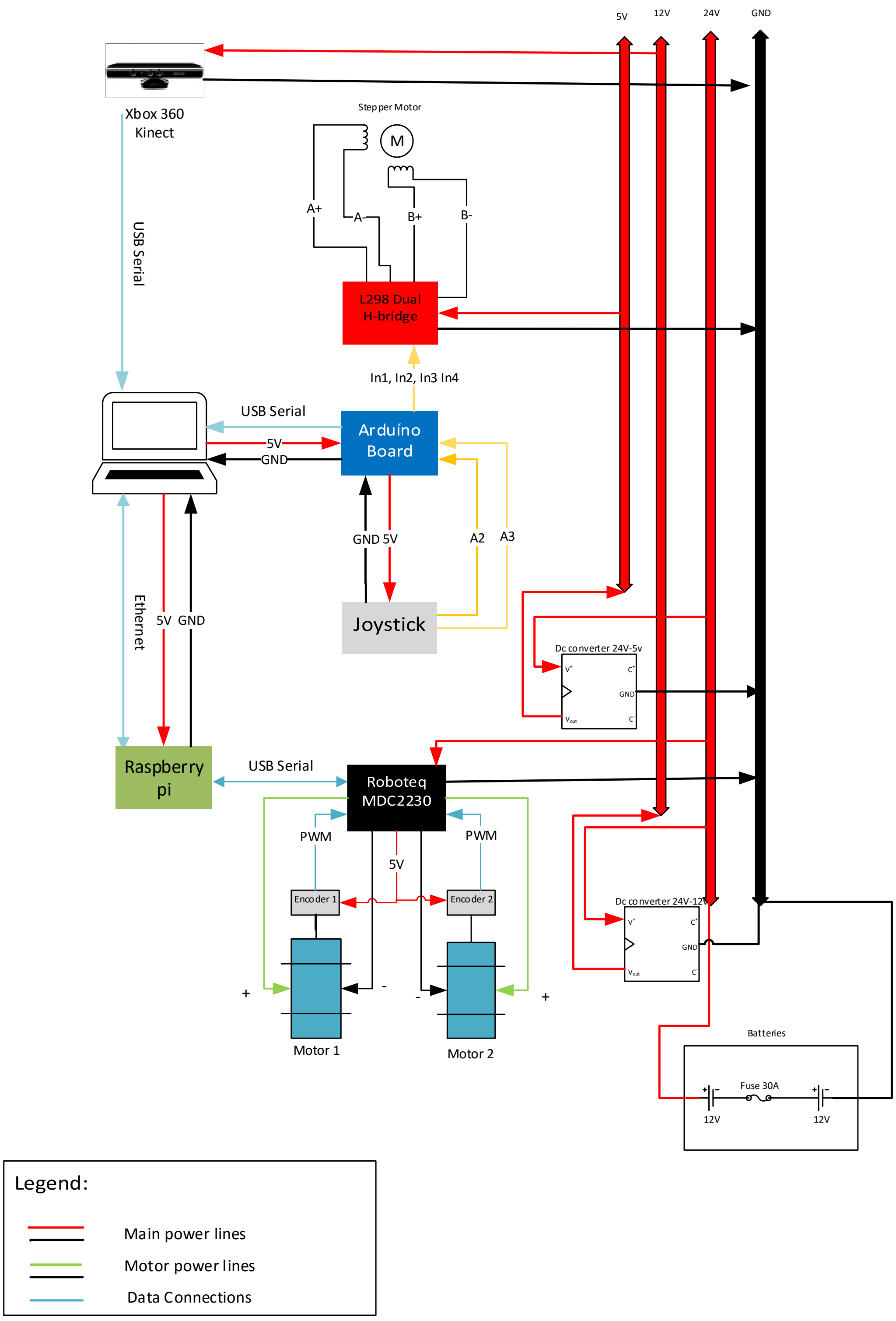


Figure 4.2: Power connections of the devices presented in RobChair Storm

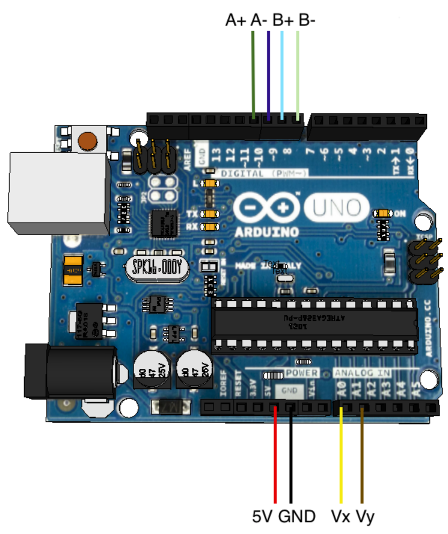


Figure 4.3: Arduino Board and connections to Joystick and Stepper Motor

The user command and the position of the Kinect (0-180º degrees) is sent to the Computer through Serial Port Communication. Figure 4.3 shows the Arduino board mounted in RobChair Storm and cable connections.

- Connections to Stepper Motor: Channel A^+, A^-, B^+, B^- ;
- Connections to Joystick: 5V, Vx, Vy and Ground (GND);

4.1.1.2 Kinect Xbox

The Xbox Kinect is a 3D depth sensor with two cameras and a laser-based infrared projector. One camera is a standard RGB camera and the other is an infrared camera. Kinect's structure can be seen in Fig. 4.4. The Kinect was chosen to provide RobChair Storm 3D or 2D perception of the environment. It has some features that make it a possible alternative to laser range finder:

- Low cost solution (about 150€)
- Works at real-time frequency (30Hz)
- Gathers 3D information of the world

However one of its major drawbacks is its operation range, that goes from 0.6 to 5m. Kinect has a blind spot which can be a problem while navigating in a dense environment, specially if safety is a critical navigation goal. To mitigate that problem the Kinect was mounted in the rear top of the wheelchair. Figure 4.5 shows the position of the Kinect on RobChair Storm.

4.1.1.3 Rotational base

To rotate the Kinect to the desired position with the Stepper Motor a rotational base that connects the stepper, and has a support platform for the Kinect, was constructed with the aid of a 3D printer. Figure 4.6 illustrates the physical platform.

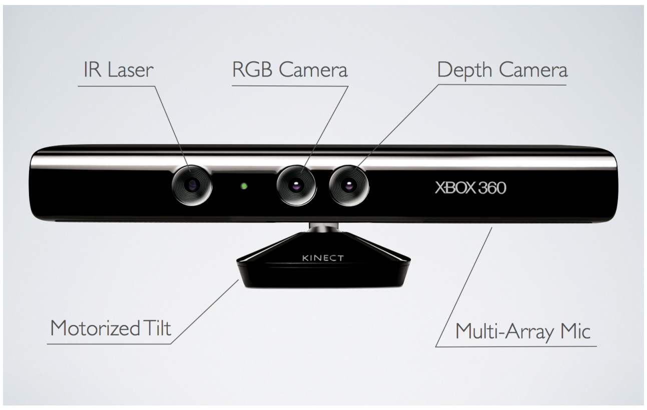


Figure 4.4: Kinect's Structure

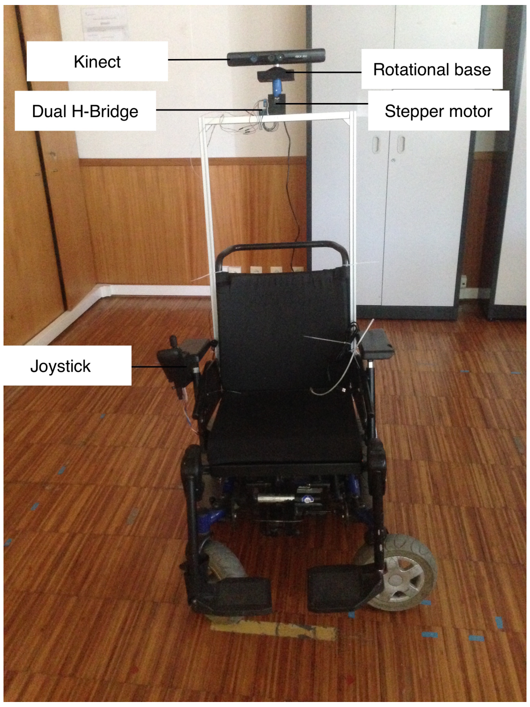


Figure 4.5: RobChair Storm

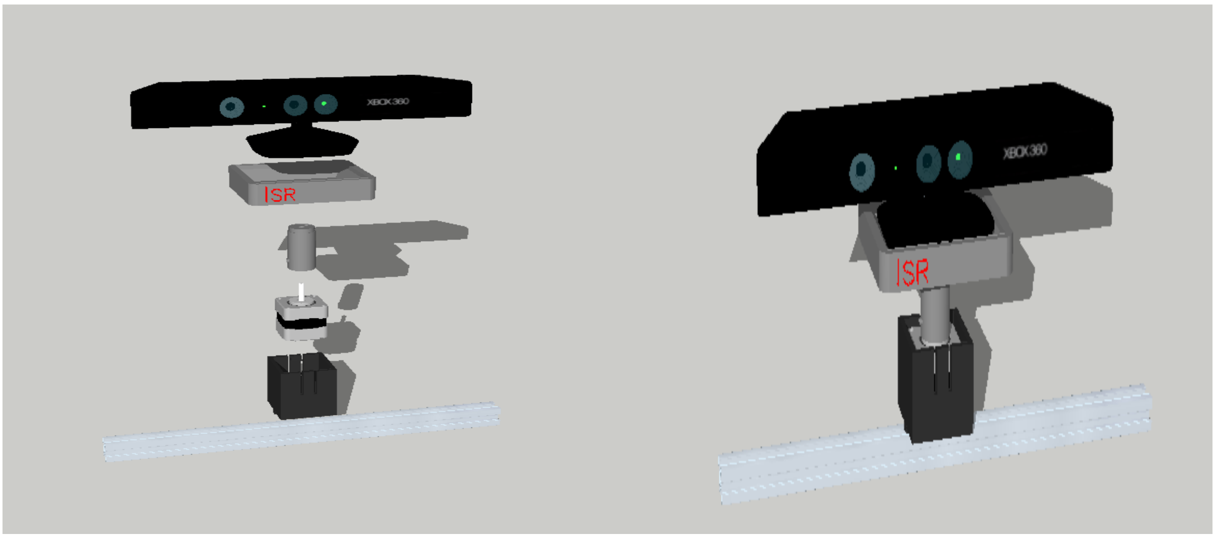


Figure 4.6: Rotational Platform to rotate Kinect

4.1.1.4 MDC2230 Motor Controller From Roboteq

The MDC2230 motor controller from Roboteq (see Fig. 4.7) is a controller that has a dual channel setup, which makes it possible to control the left and right motor of RobChair Storm individually. The controller features a high-performance 32-bit microcomputer and quadrature encoder inputs to perform advanced motion control algorithms in open loop or closed loop (speed or position) modes. The communication with MDC2230 motor controller can be done trough RS232, CAN or USB where USB is the one used to exchange data between the motor controller and the Raspberry Pi.

Figure 4.7: MDC22xx Motor Controller from Roboteq [Rob, 2015]

4.1.1.5 Raspberry Pi

The Raspberry Pi is an ARM-based single board-computer, it runs Linux as Operating System. Its task is to communicate with motor controller through USB and the Computer through Ethernet as it can be seen in Fig. 4.2. It receives all the information provided by the motor controller such as the Encoders data, batteries capacity, actual speed and others. It also sends information to the motor controller, which is mainly the desired speed for each motor, in RPM. The code and a more detailed information about how the communications are done is documented in [Gonçalves, 2013].

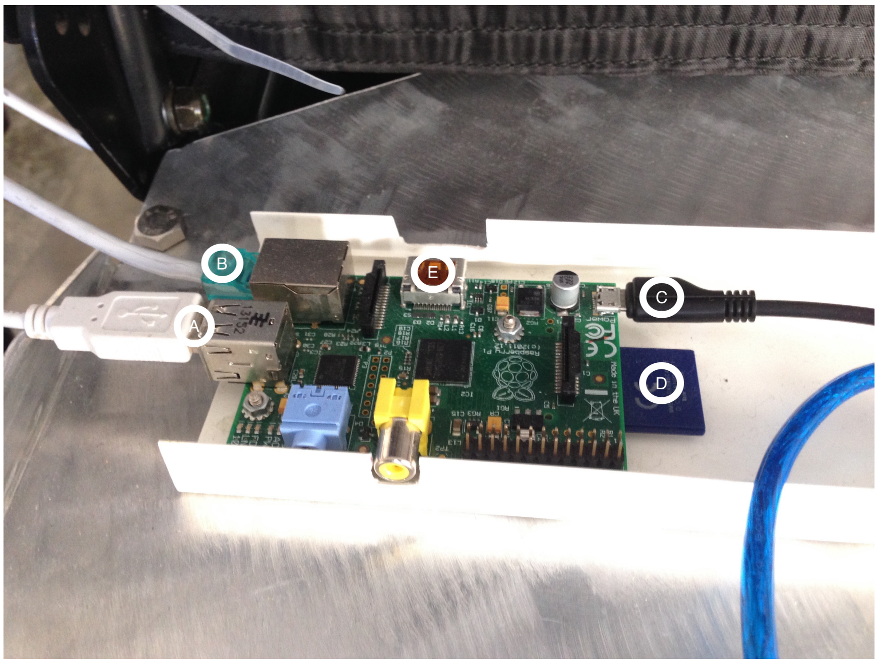


Figure 4.8: Raspberry Pi in RobChair Storm

The information is all processed in the High level Computer where the software main system is running in ROS. Figure 4.8 shows the Raspberry Pi mounted in RobChair Storm and the various components identified as:

- USB cable connected to MDC2230 (Letter A);
- Ethernet cable connected to Computer (Letter B);
- Micro USB Power Input (Letter C);
- SD card (Letter D);
- HDMI (Letter E);

4.2 PID Controller

The method used to design the PID consisted in applying a Step input in open loop mode with the RobChair Storm on the floor and retrieving the motors response. *"Ziegler and Nichols showed how optimum controller parameters could be chosen based first on open-loop tests on the plant; and second on closed-loop tests on the plant."*[Bennett, 1993].

This method already considers the friction forces included in the motor responses.

Because the transfer function is a 1st order, a PI controller is enough to eliminate the steady state error. Having this in consideration a PI controller was designed instead. To obtain the transfer function of the motor response a step of 30% of the max Voltage (max speed) was applied to each motor separately.

4.2.1 1st Order Transfer Function

After applying a 30% max speed to the motors to obtain the first model of a system, the first order transfer function of motor response is:

$$G_s(s) = \frac{G_{DC}}{\tau s + 1} \tag{4.1}$$

where G_{DC} is the DC gain and $\tau = 0.632 \cdot RiseTime$.

To validate the 1st order transfer function that models the motor a simulink diagram was designed Figure 4.9.

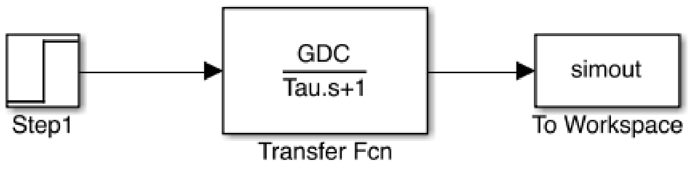


Figure 4.9: Simulink Diagram for 1st order transfer function of RobChair Storm's Motors

Processing all data retrieved in Matlab, the following values were obtained:

Parameter	Value
G_{DC}	0.65
τ	0.2308

Table 4.2: 1st order model parameters

The Figure shows that the Model obtained follows accurately the real motor response for the same input command, discarding the noise. It is also important to see that when applying a Step input to the motors in open loop the power driver can't get even close to the Reference of 300 RPM being that the reason for such lower G_{DC}.

4.2.2 PID Gains

A second order system is give by (see 4.2):

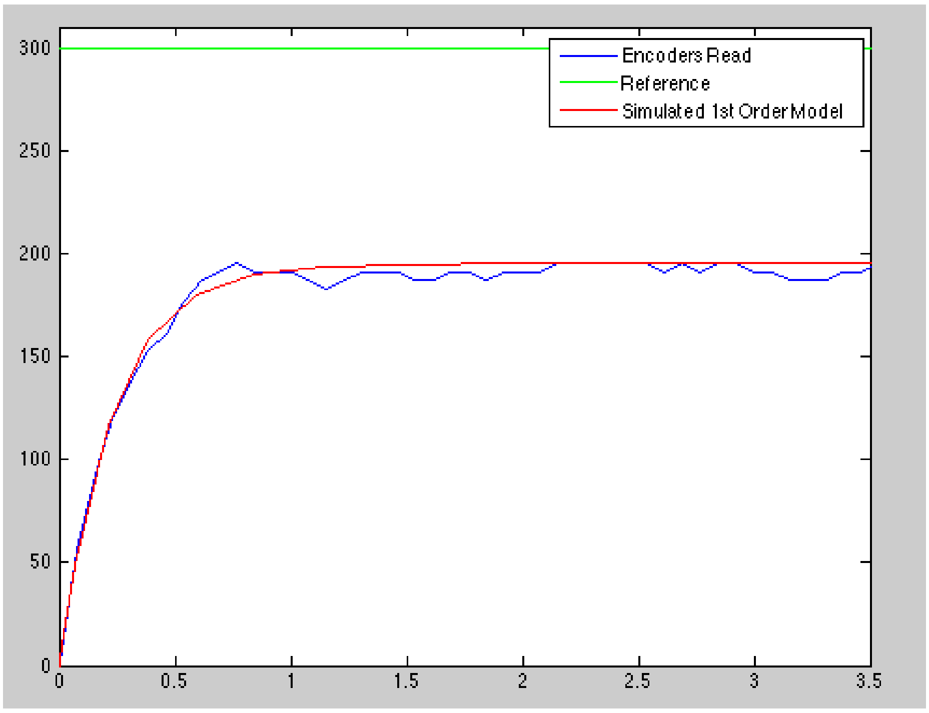


Figure 4.10: Motor response for a 30% maximum speed input command

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \tag{4.2}$$

Where ζ is the system damping determines the response shape and w_n the natural frequency represents the speed of the response.

To calculate K_p and K_i it necessary to match the transfer function in 4.2 with the closed loop transfer function represented by the simulink diagram (see Fig. 4.11)

The closed loop transfer function of the system represented in 4.11 is given by:

$$C(s) = G(s)E(s) \tag{4.3}$$

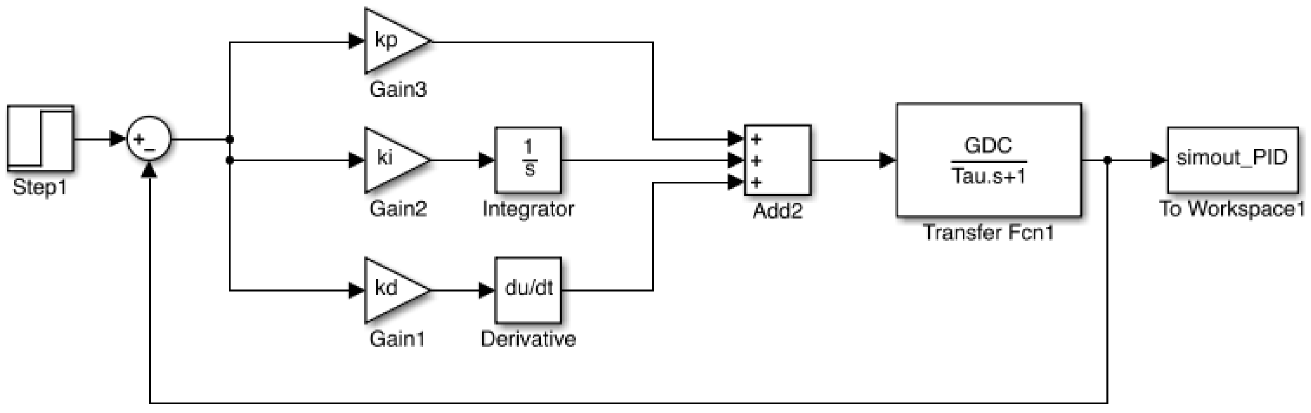


Figure 4.11: Simulink Diagram for simulating the PID Controller applied to first order model of RobChair Storm Motor

$$E(s) = R(s) - B(s) \tag{4.4}$$

$$E(s) = R(s) - H(s)C(s) \tag{4.5}$$

$$C(s) = G(s)[R(s) - H(s)C(s)] \tag{4.6}$$

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)} \tag{4.7}$$

where $R(s)$ is the reference input signal, $C(s)$ is the output, $G(s)$ is the system transfer function, $H(s)$ is the feedback element, $B(s)$ and $E(s)$ is the error between $R(s)$ and $B(s)$ (equation 4.4). To obtain the required PID gains, it is necessary to calculate the closed loop poles of the system:

$$1 + G(s)H(s) = 0 \tag{4.8}$$

$$G(s) = G_c(s)G_S(s) \tag{4.9}$$

with $H(s) = 1$,

$$1 + G_c(s)G_S(s) = 0 \tag{4.10}$$

where $G_c(s)$ is the PID transfer function that is given by:

$$G_c(s) = \frac{K_D s^2 + K_p s + K_I}{s} \tag{4.11}$$

Now replacing 4.11 and 4.1 in 4.10,

$$1 + \frac{K_D s^2 + K_p s + K_I}{s} \frac{G_{DC}}{\tau s + 1} \tag{4.12}$$

Simplifying 4.12,

$$(G_{DC}K_D + \tau)s^2 + (G_{DC}K_P + 1)s + G_{DC}K_I = 0 \tag{4.13}$$

Now it is possible to calculate the equation, which gives the PID gains, for that the equation of a second order system, 4.2 is matched with 4.13:

$$K_p = \frac{2\zeta w_n - 1}{G_{DC}} \tag{4.14}$$

$$K_I = \frac{w_n^2}{G_{DC}} \tag{4.15}$$

$$K_D = \frac{1 - \tau}{G_{DC}} \tag{4.16}$$

As referred before $K_D$was set to zero. Because this is a system for assisting people with physical

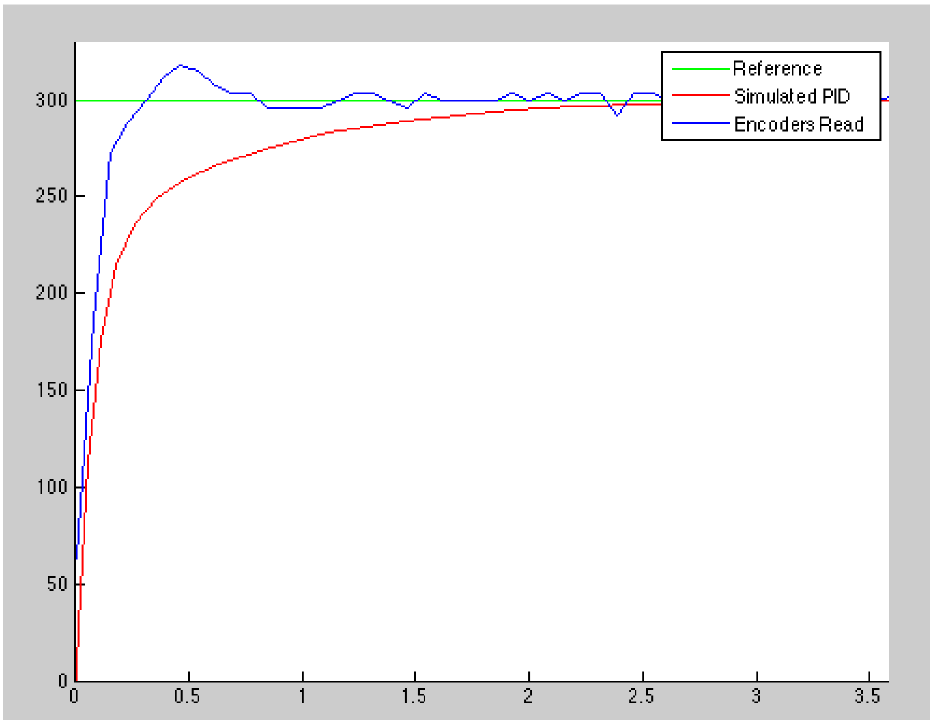


Figure 4.12: Simulated Motor response and real motor response with a PI controller

impairments the system must not have sudden movements. For that a under-damped system ($\zeta < 1$) was designed with $\zeta = 0.8$ and a natural frequency of 2 $rads/sec$.

Gains obtained:

$$K_P = 6.1 \tag{4.17}$$

$$K_I = 3.4 \tag{4.18}$$

Both motors had the same response, which gave the same gains.

4.2.3 PI Validation

This gain values are set into the RoboteQ motor controller software and the control scheme was changed from open loop mode to closed loop speed control mode.

It was applied the same input command in the real test as in the simulation on the RobChair Storm motors, taking in consideration that the encoders reads are from the RPM of the shaft and that the motors are coupled to gearboxes with 29:1 reduction factor (twenty-nine turns on the shaft for one wheel turn), this gives an input command of 10 RPM on the wheels, the Figure 4.10 shows the Simulated Response of the motor in red and the real motor response in blue. It took 0.4s to be at 300 RPM in the shaft, corresponding to 10 RPM in the wheels. This is a smooth response like it was desired. The real response has an overshoot that was not presented in the simulated response. The main reason for this is that the PI controller inserts a zero to compensate the pole in the origin. The greater the K_I gain the greater the overshoot will be. When increasing the K_I gain, the zero will start to move away from the pole in the origin, increasing the instability of the system.

Chapter 5

HMI: Joystick Modules

According to the results of the questionnaire presented in [Ana Carvalho, 2014] the joystick is still the most used HMI in PW. In this chapter the modules created to use the Joystick as HMI will be explained. These modules were designed to facilitate the RW navigation in a safe manner but maintaining the joystick hardware. Figure 5.1 illustrates the joystick module taxonomy. The latter is constituted by two main classes the Manual Mode and Semi-Autonomous Mode. The semi-autonomous mode includes, in this case, the safe navigation by using one of the following modules:

- General User intent: Computes a goal from the joystick commands (v, w), and through local navigation avoid obstacles.
- User intent with hand tremor filter: Works in the same way as the previous one, except that the user command is filtered using a Kalman filter, its usability is for users with hand tremor problems.
- User intent with discrete driving mode: In this case the RobChair Storm can be operated through discrete commands.

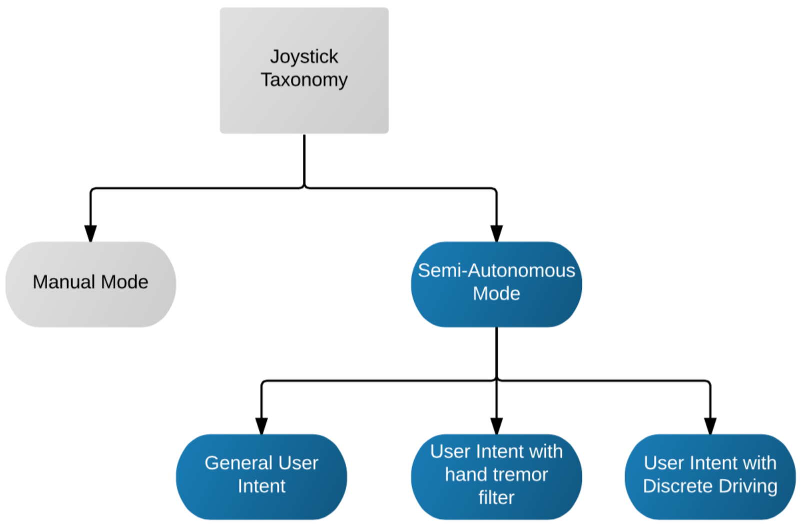

Joystick Taxonomy
Manual Mode
Semi-Autonomous Mode
General User Intent
User Intent with hand tremor filter
User Intent with Discrete Driving

Figure 5.1: Joystick Modules taxonomy

5.1 Manual Mode

The Arduino module provides the data, voltage values, to a ROS node. That data describes the interaction of the user with the joystick, where a movement along the V_y axis (see Fig. 5.2) is used to calculate the linear velocity, and a movement along the V_x axis will be used to calculate the angular

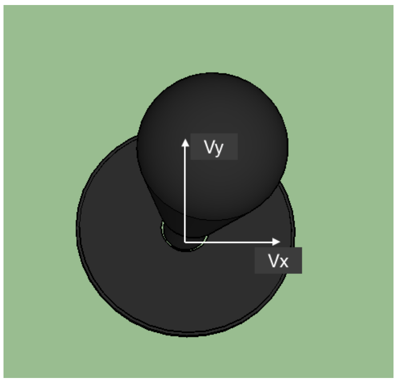


Figure 5.2: Joystick Illustration and his Axis

velocity. In an intuitive way V_y controls the back and forward movements while V_x controls the left and right rotations.

- To convert the V_y voltage values, equation $v = 2 \cdot V_y$ is used, which is represented in the Fig. 5.3 in red. The voltage values had to be centered in the cartesian system (-0.5 to 0.5V) and the linear velocity was restricted to (-1 to 1 m/s).
- To convert the V_x voltage values the same restrictions are applied. In this case the equation is $w = -2 \cdot V_x$. Since a positive angular velocity corresponds to a rotation counter clockwise the decline is negative.

In the Module Manual Mode the ANS is not used, the Linear and Angular Velocity are sent directly to the Raspberry Pi.

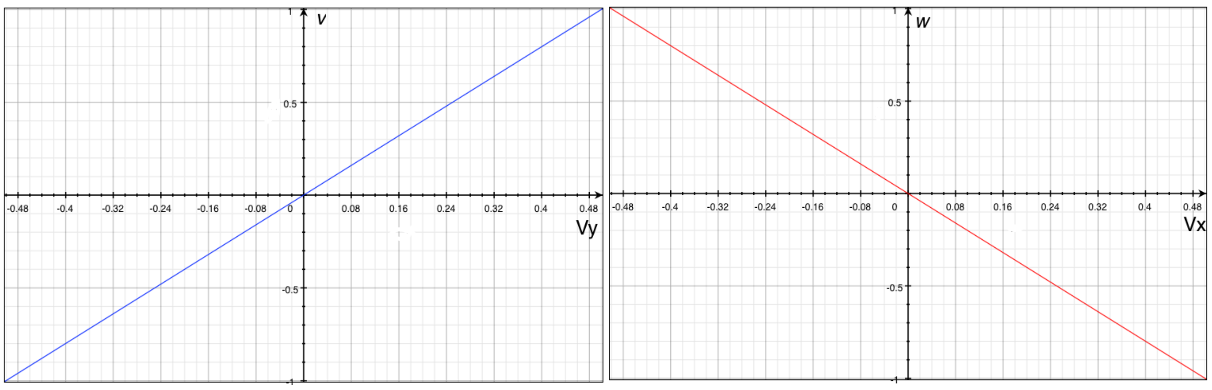


Figure 5.3: Blue line is the relationship between the linear velocity and V_y, Red line is the relationship between the angular velocity and V_x

5.2 Semi-autonomous mode

Linear and angular velocity (v, w) are used to calculate the goal representing the user intent. But before introducing how the goal is calculated, it is necessary to provide some details about the kinematic model of a differential drive robot.

5.2.1 Kinematic model of robotic wheelchairs

The RobChair Storm is constituted with two types of wheels, motorized and caster wheels. The caster wheels do not impose any restriction on the control motion of the wheelchair, this way the kinematic model of a RobChair Storm can be simplified as a differential robot. The control variables of a robot with a mechanism of differential drive are the angular velocities of motorized wheels. Let w_E and w_D be the angular velocities of left and right wheels, respectively, and R the ideal wheel's radius, the correspondent linear velocities can be calculated with $v_E = Rw_E$ and $v_D = Rw_D$. Linear and angular velocities are represented by:

$$v = \frac{R\cdot(w_D+y_E)}{2},\ w = \frac{R\cdot(w_D-w_E)}{b}$$

Where b is the distance between wheels. Knowing the linear velocity (v) and angular velocity (w), the velocities of each wheel are given by the relations:

$$w_E = \frac{v - (\frac{b}{2})\cdot w}{R} \tag{5.1}$$

$$w_D = \frac{v + (\frac{b}{2})\cdot w}{R} \tag{5.2}$$

The evolution of the characteristic variables of the model (Configuration Space R^3) can be expressed through the following relations:

$$\dot{x} = \frac{R\cdot cos(\varphi(t))}{2}w_E + \frac{R\cdot cos(\varphi(t))}{2}w_D \tag{5.3}$$

$$\dot{y} = \frac{R\cdot sin(\varphi(t))}{2}w_E + \frac{R\cdot sin(\varphi(t))}{2}w_D \tag{5.4}$$

$$\dot{\varphi} = \frac{R}{b}w_D - \frac{R}{b}w_E \tag{5.5}$$

Integrating equations 5.3, 5.4 and 5.5 in order to t, it is possible to obtain the temporal evolution of robot's pose, which allows the estimation of robot's pose through time.

$$\begin{bmatrix} x \\ y \\ \varphi \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ \varphi_0 \end{bmatrix} + \begin{bmatrix} \int \dot{x}\cdot cos(\varphi(t))dt \\ \int \dot{y}\cdot sin(\varphi(t))dt \\ \int \dot{\varphi}dt \end{bmatrix} \tag{5.6}$$

The (x_0, y_0, φ_0) are the initial pose of the vehicle.

5.2.2 General User intent

The goal that represents the user intent is a position to where the robot must go, this goal is needed to make the whole ANS to perform. Without a goal given the RobChair Storm will not move.

To determinate the goal corresponding to the user intent from the linear velocity v and the angular velocity w, the angular velocity of each wheel (w_E and w_D) is calculated with the 5.2 and 5.1. Taking into account the angular velocity of each wheel the velocity in x, y and φ can be calculated with

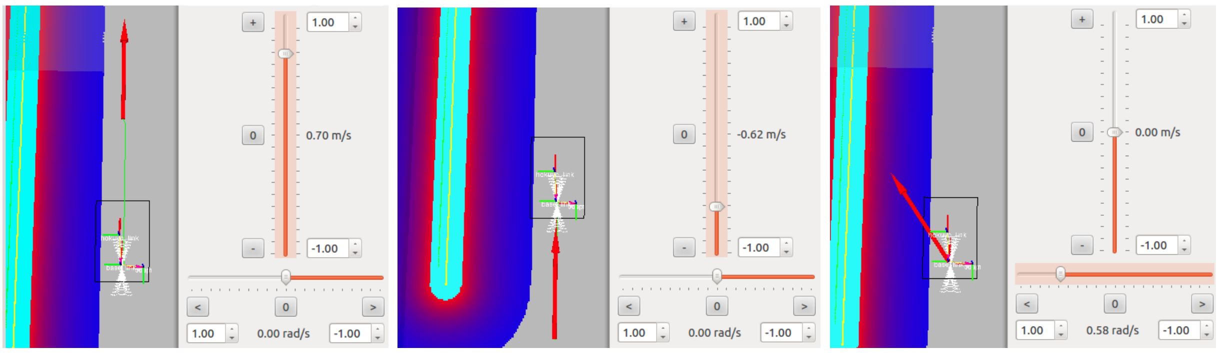


Figure 5.4: Examples of goals created from the Joystick commands: vertical slider controls linear velocities and horizontal slider controls angular velocities

5.3, 5.4, 5.5 respectively. Based on the $(\dot{x}, \dot{y}, \dot{\varphi})$ velocities it is possible to calculate a goal in (x, y, φ) by integrating 5.6, where the t is an interval from 0-3s. This approach respects the dynamics of a differential drive, so this way it is not possible to send a goal that requires to many maneuvers to reach as for example a goal in (0,1,0).

Figure 5.4 shows goals produced by linear and angular velocities. On the left is a forward goal computed only with a positive linear velocity, on the center is a backwards goal with a negative linear velocity, and on the right is pure rotation goal where only an angular velocity is used. It is also important to notice that the initial pose is ignored during the computation of each goal because it was considered that the RobChair Storm is, at any instance, at the origin.

5.2.3 User intent with Hand Tremor Filter

Users suffering some hand motion disorders find it difficult or impossible to operate a conventional joystick. Many of these users are not able to steer the conventional joystick without exhibiting large-position amplitude deviations. In this dissertation a system is proposed to mitigate any hand tremor and is unresponsive to rapid and erratic movements. To filter the user commands, the Kalman Filter [Welch and Bishop, 2006] is applied to v and w velocities from the Joystick. Algorithm 5.1 presents the Prediction and correction stage of Kalman Filter.

Two circular buffers were used, that are updated every time a new command is received, to save five samples of t v and w velocities. Using those samples, the mean $(\overline{x})$ and the covariance (σ^2) are calculated. The covariance of v and w velocities are the covariance of the measurement noise, which means that R will be:

$$R = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_w^2 \end{bmatrix}$$

As R represents the uncertainty in the measurements a great covariance represents a great uncertainty.

To determine Q, the matrix of the process noise covariance, an iterative trial and error process was used. Figures 5.5, 5.6, 5.7 show the behavior of the filter for different Q Matrixes.

Figure 5.5 shows that for a $Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$, the user command signal is already filtered, but it is not enough to really mitigate hand tremors.

Algoritmo 5.1 Kalman Filter Algorithm

```
$Q = \begin{bmatrix} 0.0001 & 0 \\ 0 & 0.0001 \end{bmatrix}$
$P_0 = Q$
$x_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
loop:
-> Prediction Stage
$\hat{x}_k^- = A\hat{x}_{k-1}$
$P_k^- = P_{k-1} + Q$
-> Correction Stage
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
$P_k = (I - K_k H)P_k^-$
End loop
end procedure
```

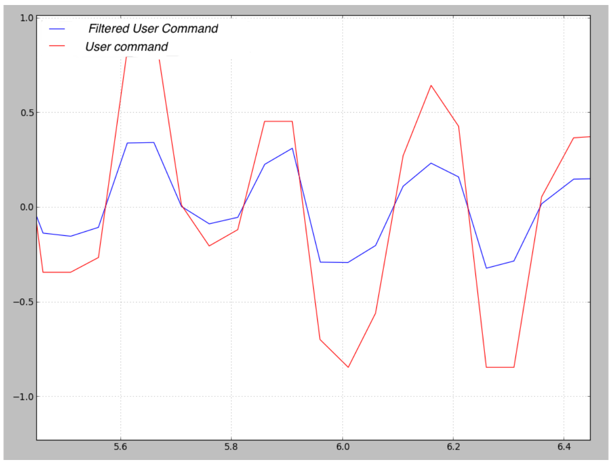


Figure 5.5: Kalman Filter with Q = [0.1 0; 0 0.1], the red wave is the user command and blue wave is the user command filtered

Giving an even less uncertainty to the system, lowering the values of Q, the filtering will work better. Figures 5.6 and 5.7 show the results for a $Q = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.001 \end{bmatrix}$ and a $Q = \begin{bmatrix} 0.0001 & 0 \\ 0 & 0.0001 \end{bmatrix}$ respectively. After analyzing both Figures it is possible to observe that the results of the filter in Fig. 5.7 shows a better performance, since it ignores completely any hand tremor while in Fig. 5.6 there is still a little response to the tremor. And from the Fig. 5.8 it can be seen that the system will not respond to a rapid movement from the user.

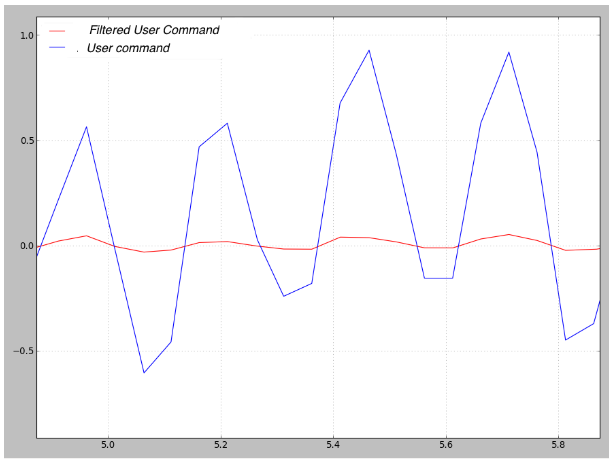


Figure 5.6: Kalman filter with Q = [0.001 0; 0 0.001]

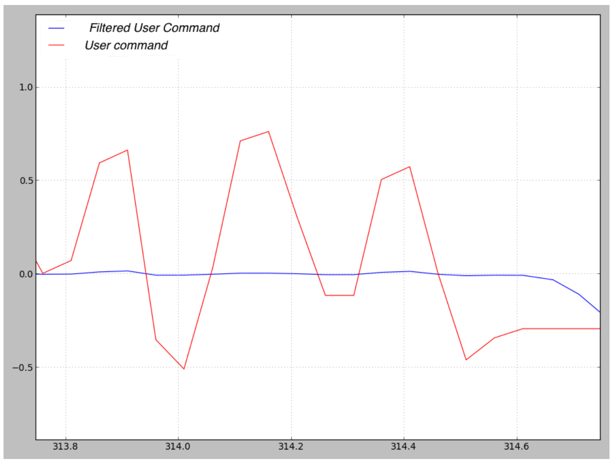


Figure 5.7: Kalman Filter with Q = [0.0001 0; 0 0.0001]

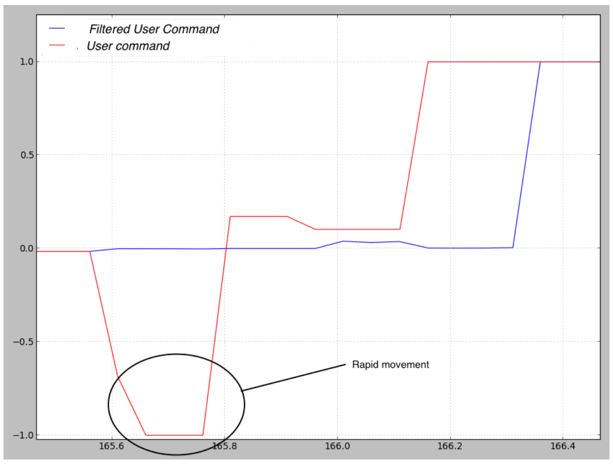


Figure 5.8: Rapid movement example, Kalman Filter with Q = [0.0001 0; 0 0.0001]

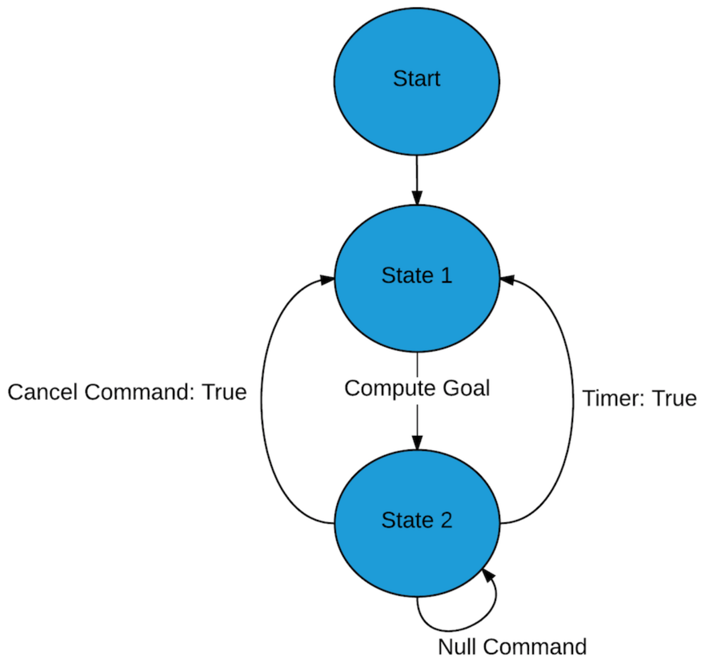


Figure 5.9: Diagram of the Discrete Driving Algorithm

5.2.4 User intent with discrete driving mode

The discrete driving solution purposed, in this dissertation, is once again a pre-condition of the user commands (v, w) through an algorithm. The discrete driving algorithm has two states, as depicted in Fig. 5.9. State 1 receives the user command and computes the goal from that command, if the user command is Null which means that both v and w are 0 the algorithm stays in state 1. State 2 is where the algorithm keeps the user command received in state 1 and keeps calculating a goal from that command, while the coming user commands are interpreted. The user command are interpreted differently depending on which state the algorithm is. It is divided into three types of commands:

- New Command: a user command is considered a “New command” if the algorithm is in the state 1, which means that any command from the user in this state will be used to compute a goal.
- Null Command: a user command is considered a “Null Command” if the algorithm is in the state 2 and the joystick is in the resting position, which means that both v and w are 0.
- Cancel Command: a user command is considered a “Cancel Command” if the algorithm is in the state 2 an the joystick is not in the resting position.

Figures 5.10 and 5.11 represent simulations carried out to better understand how the discrete driving system works. In Fig. 5.10 in blue is the user command and in red is the command that will be used to compute the goal. Three types of commands are highlighted in 5.10. It starts with the joystick in the “Resting Position” and in State 1, then a “New Command” is received, the algorithm changes from State 1 to State 2. Then the joystick goes to the “Resting Position”, where the command is considered a “Null Command”, keeping the algorithm in state 2. While the “Cancel Command” is not given the algorithm keeps the user command equal to the one received in the state 1. Which is possible to observe by looking that in state 2 the red wave is constant. When a “Cancel Command” is given the algorithm leaves the state 2 and goes to state 1.

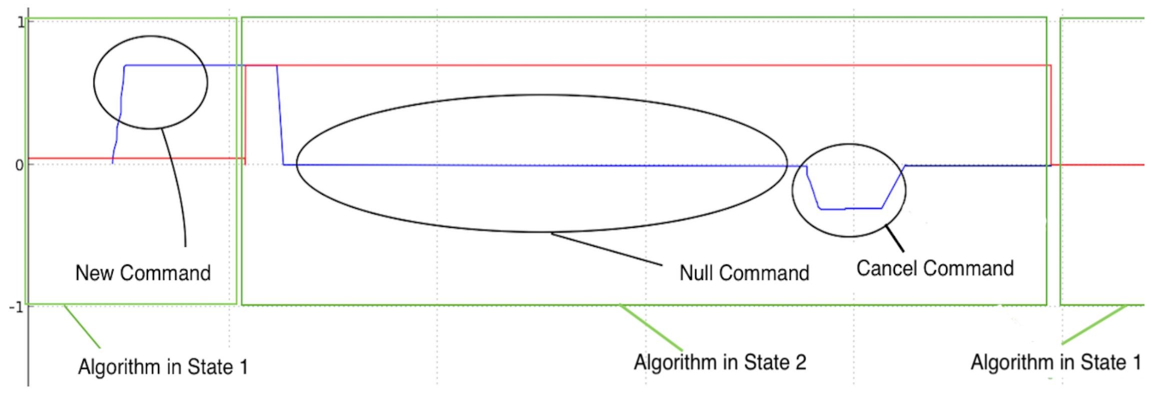


Figure 5.10: Discrete Driving Example: User Command and Cancel Command

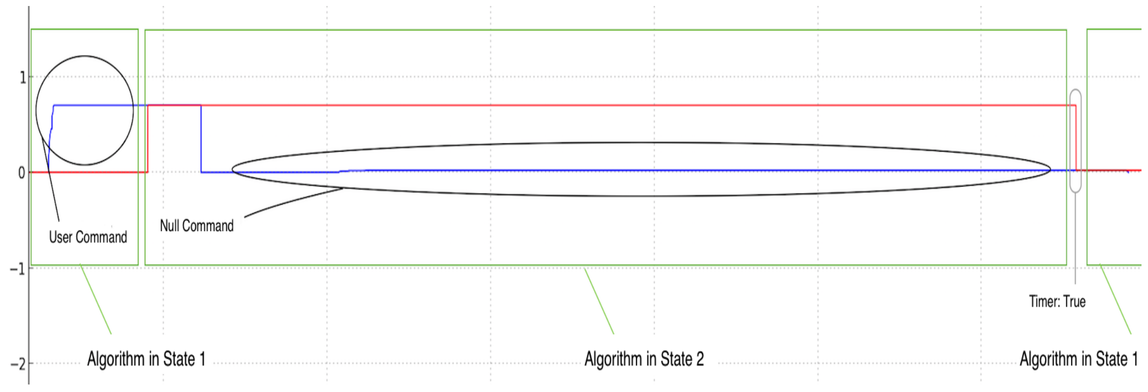


Figure 5.11: Discrete Driving Example: Command Canceled after t>10s

The user command that makes the algorithm to leave state 1 and to go to state 2, is only valid for 10 seconds. After this time the “Timer” variable becomes true and the algorithm also leaves the state 2. Figure 5.11 shows what happens if the user never gives a “Cancel Command” and the “Timer” becomes true.

Chapter 6

Tests and Results

To validate the entire system at hardware and software levels, tests were made to the KBPS and to the ANS with the various joystick modules. This chapter will embrace those tests to clarify if the system is robust, and to find out the limitations presented by the proposed approaches and methodologies. First a comparison between the costmaps built with the 2.5D fake laser scan data and 3D pointcloud was carried out.

6.1 Kinect based perception system

6.1.1 2.5D fake laser scan limitations

As referred before the 2.5D fake laser scan can delete some important information about the surrounding environment. An example of that is the scenario presented in Fig. 6.1. In this scenario, constituted by a small table and the surroundings chairs, the fake laser gives an inaccurate information of the world. It deletes the fact that behind the chair there is a table and the path is not free.

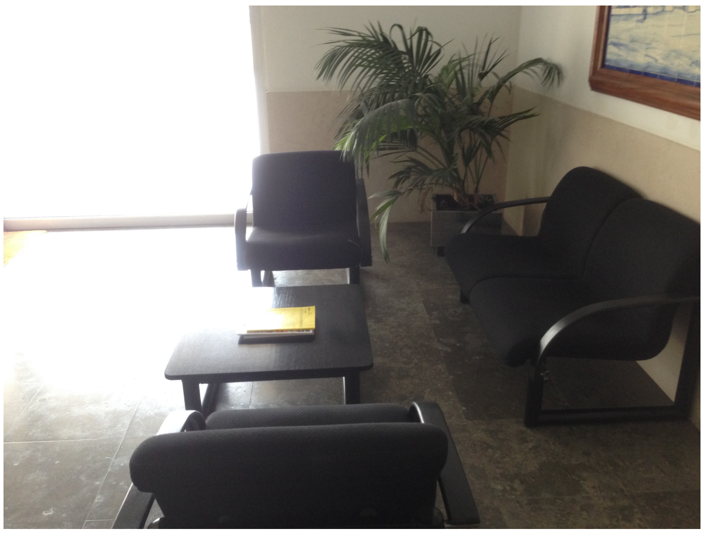


Figure 6.1: Example of a real scenario for which the 2.5D fake laser may be a problem, due to the occlusion of the table.

Therefore, the user sends a command to move forward, the fake laser based costmap will provide erroneous information to the planner, originating a path through occupied space, as it is possible to observe in the left part of Fig. 6.2. The right side of Fig. 6.2 is the costmap of the exact same scenario but created from the pointcloud. In this situation the user command is ignored and the RobChair Storm does not do anything. This shows that the 3D depth information was able to create a costmap with more accurate information of the environment.

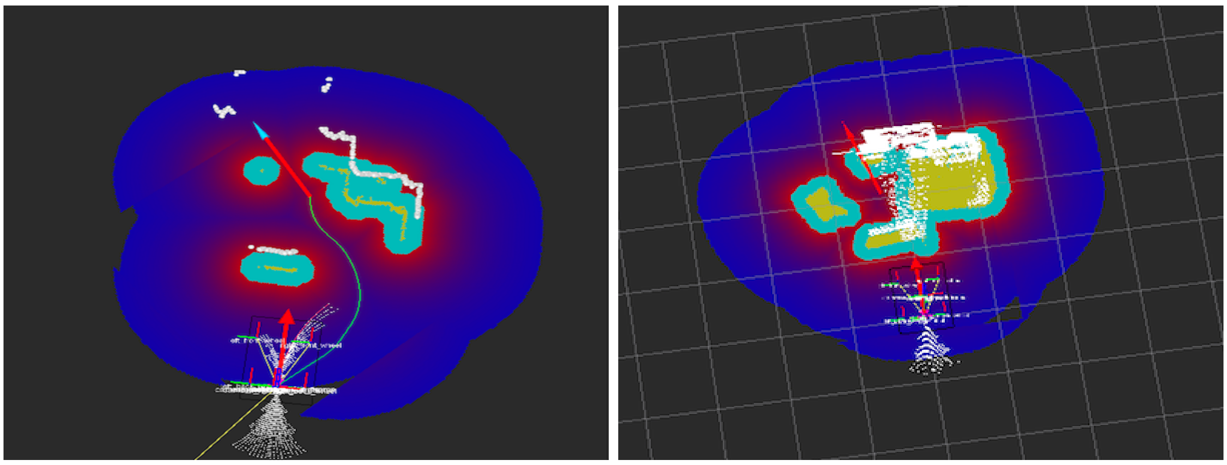


Figure 6.2: User intent of going forward in the scenario presented in Fig. 6.1 created with the laser scan (on the left) and with the pointcloud (on the right)

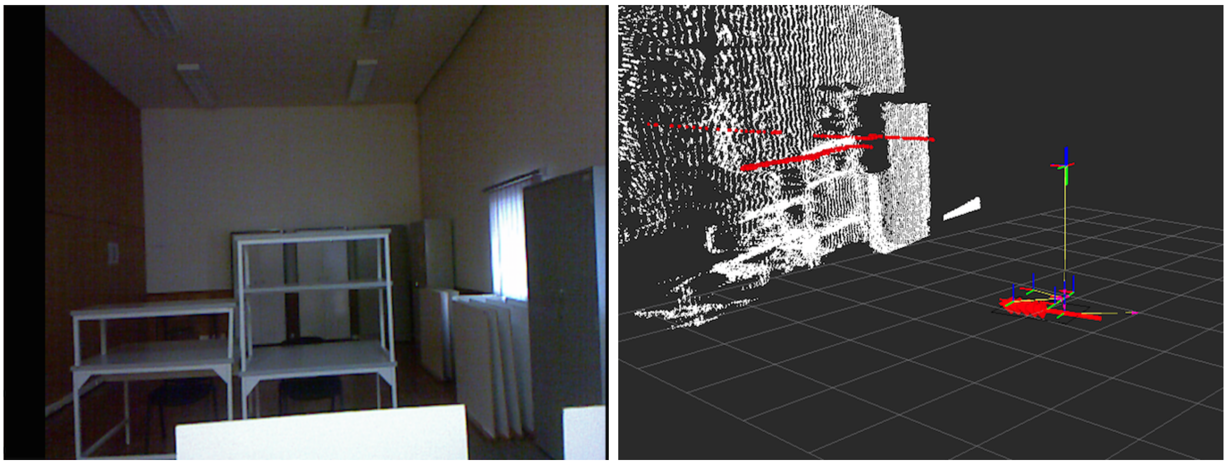


Figure 6.3: Scenario where the obstacle is not visible by the Kinect if it has a 0 degree tilt angle.

6.1.2 Kinect Visibility

In the tests presented in this sessions, the Kinect sensor was mounted with a tilt angle of -20 degrees. That was necessary because the Kinect was mounted in the rear top, and obstacles too close to the RW would not be seen, as in the scenario presented in Fig. 6.3. In this scenario the white board is not visible by the 2.5D fake laser scan.

To mitigate that, and because the Kinect has a small motor that allows the change of the tilt angle, -20 degree tilt angle was provided. Figure 6.4 shows that this problem is mitigated and the whole white board is now seen by the Kinect and is seen by the 2.5D fake laser scan.

6.1.3 Map update time

In terms of information that can be stored in the costmap the use of the pointcloud to create costmaps is better than the 2.5D fake laser scan. The problem is the computational effort to create a costmap with pointclouds. Table 6.1 has some information to compare the use of both approaches. These results were taken from a set of tests. Building and updating costmaps based on pointclouds with the available computational power may compromise the ANS safety. Taking so long time to update the map cause

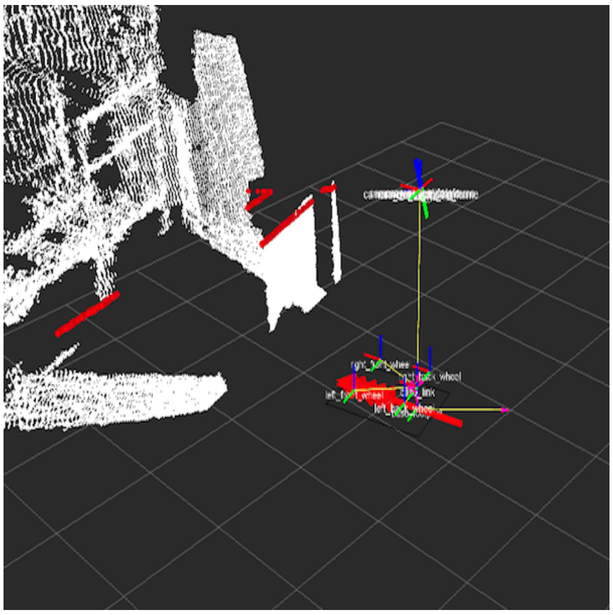


Figure 6.4: Kinect mounted with a tilt angle of -20 degrees.

significative delays in publishing the velocities messages required to make sure the system works safely and efficiently in real time, since the commands issued by the users would take an unbearable amount of time to be processed. For that reason the remaining tests presented in this chapter were carried out using the fake laser strategy.

	Minimum time to update map	Maximum time to update map	Mean time to update map
Fake laser Scan	0.10 seconds	0.77 seconds	0.2664 seconds
Pointcloud	0.6350 seconds	1.6231 seconds	1.07454 seconds

Table 6.1: Pointcloud vs 2.5D fake laser scan

6.2 ANS tests

6.2.1 ANS test in simulated environment

Before making the tests in the real platform, a simulation in ROS was performed to show how the general user intent behave. Figure 6.5 shows the path associated to the navigation task performed using the general user intent mode. As depicted in Fig. 6.6 a simple forward continuos command, provided with the joystick, was enough to move the RW from one room to another as shown in the path depicted in Fig. 6.5.

The tests made to assess the ANS consisted on testing every joystick module and evaluating their performance. Three different able-bodied users performed the tests. Subject 1 and 2 are used to navigate a PW, the third user does not have experience as the others. The scenario consisted on passing through a door way and then overcome the obstacles that appeared on the path. The scenario is depicted in Fig. 6.7.

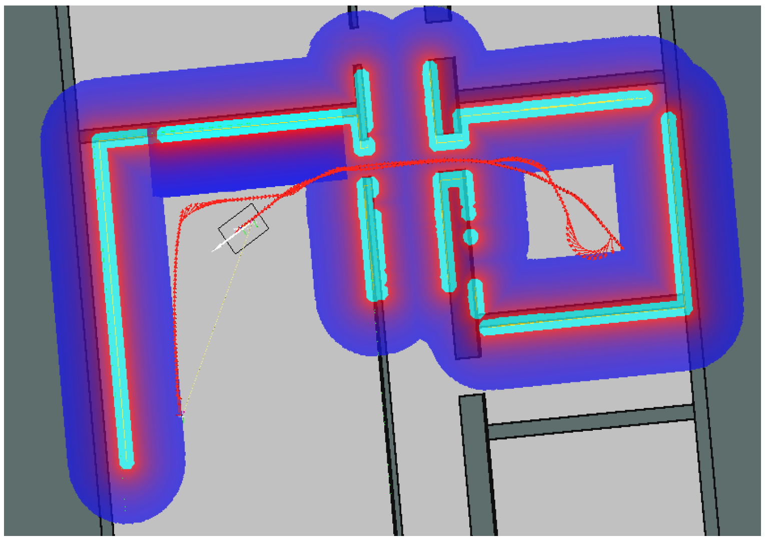


Figure 6.5: Navigation task consisting of moving the RW from one room to another.

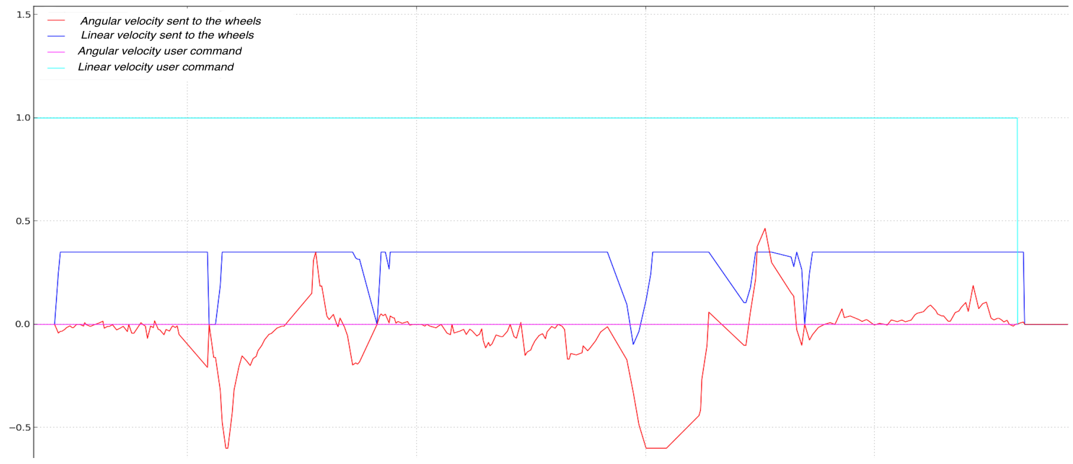


Figure 6.6: Linear and angular user commands and real velocity sent to the wheels

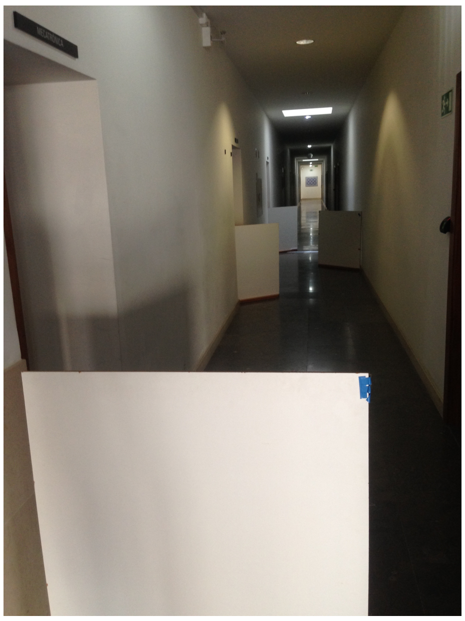


Figure 6.7: Scenario of the tests performed to the ANS.

6.2.2 Experimental cases

- Experimental Case 1: this experimental case was made to validate the "Manual" mode. To perform this test the user was asked to pay good attention to the obstacles and perform the circuit the fastest he/she could. Figure 6.8 represents the path followed by subject 1from the starting to the final point.

- Experimental Case 2: this experimental case was made to validate the "General User Intent" mode . In this test the user was asked to be always sending a forward movement. See Fig. 6.9 to observe the path followed by subject 1.

- Experimental Case 3: this experimental case was made to validate the "User Intent with Hand Tremor Filter" mode. In this test the user was asked to keep sending irregular commands along the V_x axis of the joystick and keeping a forward movement. See Fig. 6.10 for the route made by subject 1 in this test.

- Experimental Case 4: this experimental case was made to test the "User Intent with Discrete Driving" mode. In this test the user was asked to perform the circuit by providing the system with a command every ten seconds or every time he/she intends to change the direction of the movement. Figure 6.11 shows the path followed by subject 1 in this test.

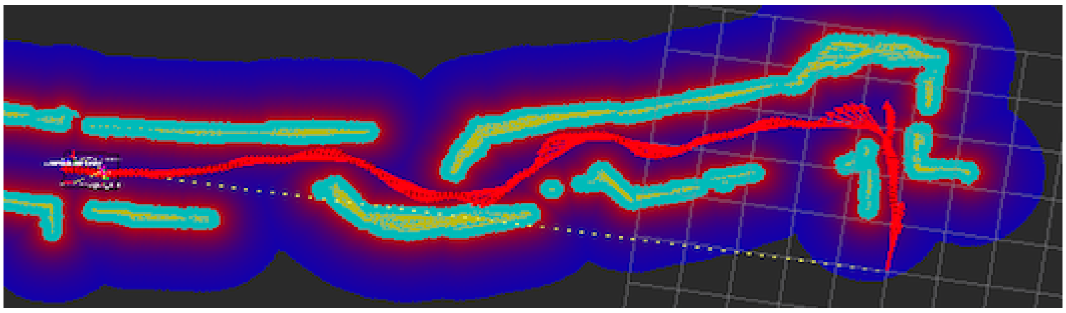


Figure 6.8: Path followed in experimental case 1

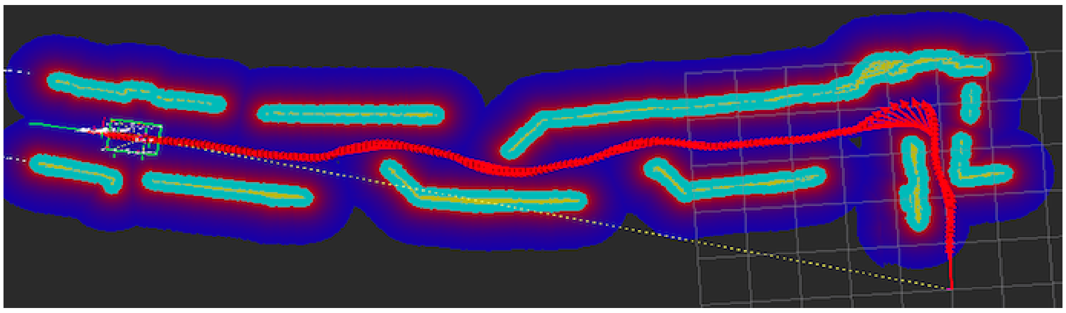


Figure 6.9: Path followed in experimental case 2

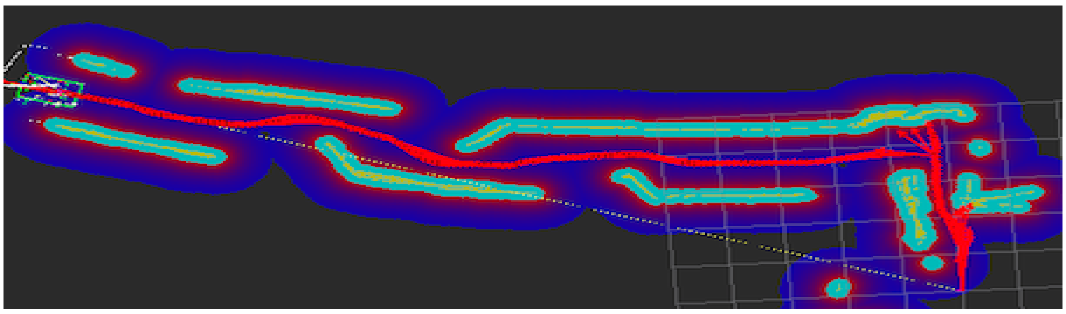


Figure 6.10: Path followed in experimental case 3

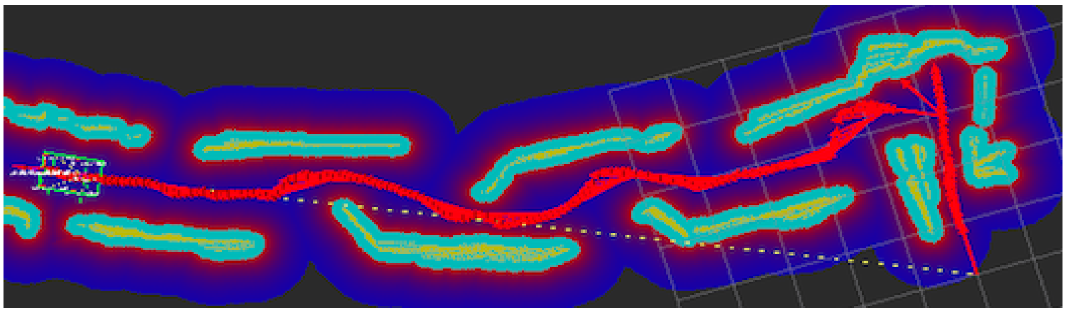


Figure 6.11: Path followed in experimental case 4

6.2.3 Backwards safety

It was also made a small experimental case to validate the safety in backwards, that is made by the rotation of the Kinect to give a backward vision. Figure 6.12 has the path that RobChair Storm did on backwards.

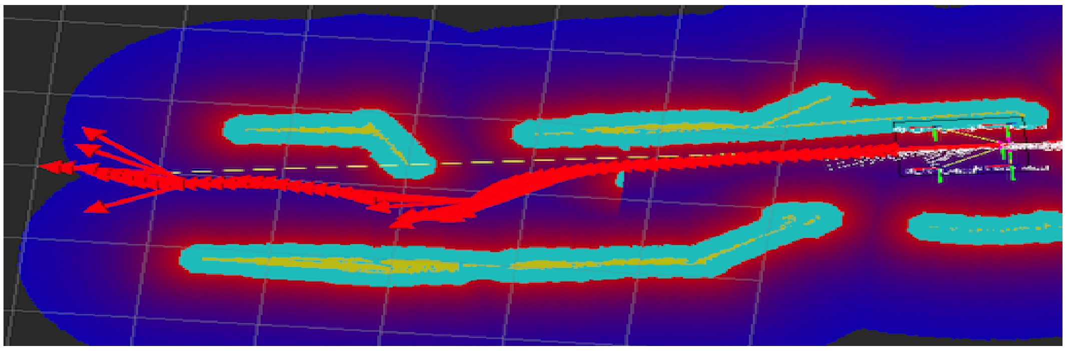


Figure 6.12: Backwards safety experimental case

6.2.4 Experimental cases discussion

	Time	Distance Traveled	Max velocity	Mean Velocity	Clearance	Collisions
Subject 1: Case 1	55s	16.42m	0.35m/s	0.29m/s	0.112m	0
Subject 1: Case 2	61s	16.31m	0.35m/s	0.26m/s	0.172m	0
Subject 1: Case 3	76s	16.61m	0.35m/s	0.22m/s	0.133m	0
Subject 1: Case 4	78s	16.15m	0.35m/s	0.20m/s	0.143m	0
Subject 2: Case 1	59s	15.97m	0.35m/s	0.27m/s	0.158m	0
Subject 2: Case 2	61s	16.43m	0.35m/s	0.27m/s	0.167m	0
Subject 2: Case 3	74s	16.37m	0.35m/s	0.22m/s	0.161m	0
Subject 2: Case 4	79s	16.10m	0.35m/s	0.20m/s	0.168m	0
Subject 3: Case 1	67s	16.71m	0.35m/s	0.25m/s	0m	1
Subject 3: Case 2	55s	16.39m	0.35m/s	0.29m/s	0.184m	0
Subject 3: Case 3	54s	15.90m	0.35m/s	0.29m/s	0.162m	0
Subject 3: Case 4	63s	16m	0.35m/s	0.26m/s	0.157m	0

Table 6.2: Metrics obtained for the experimental cases

Table 6.2 shows the most relevant metrics of the experiments carried out with the three subjects.

- Subject 1 results: Subject 1 had no troubles on finishing the navigation task in manual mode. However the subject performed close approaches to the obstacles (0.10m). The clearance to obstacles has been improved with the proposed ANS. However, the user took more time to finish the task, once in ANS modes the user is not allowed to accelerate and brake as in manual mode. Additionally, the performed path was significantly smoother.

- Subject 2 results: Subject 2, compared to subject 1, had a similar performance. The time to complete the navigation tasks got worse compared to the “Manual Mode”, even though the difference of time is not very significative.

- Subject 3 results: Subject 3 was the subject with a lot less train on using PW and that was seen while transversing the door way, when a soft collision occurred while maneuvering the PW. When performing with the ANS activated, this subject had much better results than the previous ones. This situation probably happened because this user had no experience driving the PW, and for that reason it followed the instructions provided. This user took less time to perform any ANS tasks when compared with manual mode.

For experienced users the ANS somehow hinder the performance on navigation tasks, a reason for that is, that the ANS reduces the speed of the wheelchair when in the presence of obstacles which is reasonable since it provides a safer navigation this way. On the other hand the levels of concentration needed to navigate with the ANS are less. The user might even do other tasks while navigating or concentrate his vision in the surrounding environment.
While for users with less experience, the ANS show that it could be useful since it improved the navigation experience by avoiding collisions, improved the time and the mean velocity compared to the manual mode and had also better results than the other subjects when using the ANS.

Having the subject 3 as example, because it was the one with the best performance with the ANS, it is possible to understand that the “User Intent with discrete driving mode” is somehow slower than

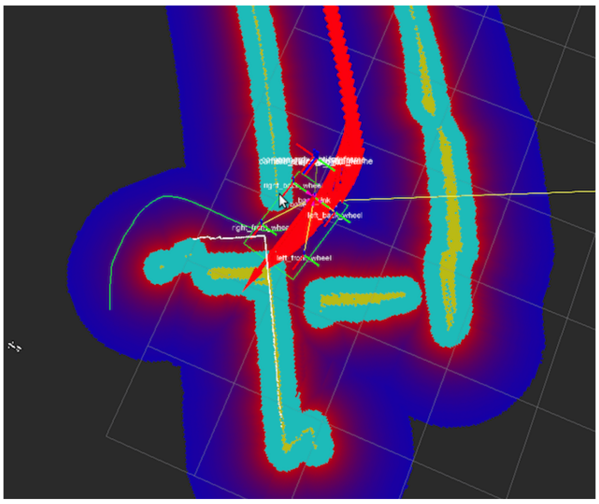


Figure 6.13: Problem of passing a door way

the other modes, that's because of the ten seconds validation which command has. On the other hand it becomes a lot less tiring to use. Being that the fact that the user do not have to give a continuous command to navigate and still was able to finish the path in a good time.

6.3 Limitations of the ANS

One of the limitations of the purposed ANS is concerned with planning failures during navigation of narrow spaces, such as door passage. For instance Fig. 6.13 is an example of a situation of this type. In this case the user has the intention of going to the division on the right, as it can be seen by the global path (green line). DWA in this circumstances fails because it is intended for small differential circular robots, and shows some limitations for big rectangular shape robots. It will also have problems during rotations in complex situations with obstacles nearby, even if the distance allows the rotation.

Another problem is that the move_base package proved it self to be really heavy at computational levels. This caused small problems when the computer was not plugged in, which decreases the computational power available. When this happens the publication rate of velocities messages would decay to 1-2Hz leading to rough stops. This behavior is not observed if the computer is plugged in.

The Kinect with a tilt angle can see almost every obstacle in the path if the object is not too small and is already too close to the RobChair's front. Otherwise the latter is not detectable. This is a problem in an office, for example, where a lot of small objects can be on the floor. The consequences of this problem is still less than the use of the Kinect in the front of RobChair Storm, reminding that Kinect has a blind spot behind 60cm.

Chapter 7

Conclusions and future work

7.1 Conclusions

In this dissertation a PW was redesigned and an ANS was developed to provided safety while using the RW. The ANS is implemented in a real platform, RobChair-STORM, which was completely restructured. This assistive system is showing good results despite the fact that sometimes when passing narrow spaces like door ways, it may fail on accomplishing the user intent. The Kinect is used as sensor to give perception of the environment. However, the Kinect sensor presents some major drawbacks related to its blind spot, high computational complexity if the pointcloud is directly used to construct the costmap, and building of inaccurate costmaps when the fake laser strategy is used, which may compromise safety.

In this dissertation, the development of the electric and electronic system required by a robotic wheelchair was developed and evaluated. An assistive navigation system based on the information provided by a Kinect sensor was also developed and implemented. A new HMI module based on a joystick was proposed. This module is able to operate in different modes, namely: general user intent, user intent with hand tremor filtering, and discrete driving mode for users incapable of providing a continuos steering command.

From the most valued features, limiting factors and possible improvements in a powered wheelchair that were pointed out in [Ana Carvalho, 2014] it is possible to concluded that:

- The HMI joystick-based modules provide an easy navigation of the RobChair Storm;
- Safety was a major concern and was partially achieved;
- The dimensions of the PW were not modified, or increased by the additional sensors required by the RW platform;
- Aid in reverse drive was achieved by rotating the Kinect, giving perception of the environment on the back of the RW;
- The joystick is proposed as HMI, which is the most common interface being used in powered wheelchairs;
- Obstacle avoidance was implemented and tested;

7.2 Future work

In the future, and because the ANS was implemented to support people with motor impairments, ANS should be validated by users suffering from severe motor impairments. To avoid the limitations caused by the move-base ROS navigation stack, a new planning approach that takes into account the geometry of the RW should be developed and implemented. Additionally, and to avoid the major drawbacks caused by the use of a Kinect sensor, additional information provided by other sensors, such as laser scanner and/or sonars, is required.

Bibliography

[Wik, 2013] [online](2013) [cited 2015]. Available from: `https://en.wikipedia.org/wiki/RANSAC`.

[Rob, 2015] [online](2015). Available from: `http://www.roboteq.com/index.php/roboteq-products-and-services/brushed-dc-motor-controllers/mdc2230-detail`.

[PCL, 2015] [online](2015). Available from: `http://docs.pointclouds.org/1.7.0/group__sample__consensus.html`.

[dep, 2015] [online](2015). Available from: `http://wiki.ros.org/depthimage_to_laserscan`.

[RAN, 2015] How to use random sample consensus model [online]. (2015). Available from: `http://www.pointclouds.org/documentation/tutorials/random_sample_consensus.php`.

[Ana Carvalho, 2014] Ana Carvalho, Alireza Asvadi, C. C. A. L. U. N. (2014). Mobility, accessibility and safety of people with cerebral palsy. *AMS- HMI2012: Assisted Mobility Supported by shared control and advanced Human-Machine Interfaces.*

[Aylor et al., 1979] Aylor, J., Ramey, R., Schaddegg, J., and Reger, S. (1979). Verstile wheelchair control system. *Med. Biol. Eng. Comput.*, 17:110–114.

[Baklouti et al., 2010] Baklouti, M., AbouSaleh, J., Monacelli, E., and Couvet, S. (2010). *Human Machine Interface in Assistive Robotics: Application to a Force Controlled Upper-Limb Powered Exoskeleton.* Number 978-953-7619-78-7. InTech, houssem abdellatif edition.

[Bonarini et al., 2012] Bonarini, A., Ceriani, S., Fontana, G., and Matteucci, M. (2012). Introducing lurch: a shared autonomy robotic wheelchair with multimodal interfaces. *IROS 2012 Workshop on Progress, challenges and future perspectives in navigation and manipulation assistance for robotic wheelchairs.*

[Brenner et al., 2001] Brenner, C., Haala, N., and Fritsch, D. (2001). Towards fully automated 3d city model generation. *In Automatic Extraction of Man-Made Objects from Aerial and Space Images III.*

[Dijstra, 1959] Dijstra, E. W. (1959). A not on two problems in connexion with graphs. *Numerische mathematik.*

[Fattouh et al., 2004] Fattouh, A., Sahnoun, M., and Bourhis, G. (2004). Force feedback joystick control of a powered wheelchair: Preliminary study. *IEEE Int. Conf. S.M.C.*

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM.*

[Fox et al., 1997] Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *Robotics Automation Magazine.*

[Gerkey and Konolige, 2008] Gerkey, P. B. and Konolige, K. (2008). Planning and control in unstructured terrain. *In Workshop on Path Planning on Costmaps if the IEEE International Conference on Robotics and Autonmation.*

[Gonçalves, 2013] Gonçalves, D. (2013). Robchair 2.0: Simultaneous localization and mapping and hardware/software frameworks. Master's thesis, University of Coimbra.

[Grasse et al., 2010] Grasse, R., Morere, Y., and Pruski, A. (2010). Assisted navigation for persons with reduced mobility: path recognition through particle filtering (condensation algorithm). *Journal of Intelligent and Robotic Systems*, (60):19–57.

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of a minimum cost paths. *Annals of Operations Research.*

[Hillman and Jepson, 1992] Hillman, M. and Jepson, J. (1992). Evaluation of a robotic workstation for the disabled. *Journal of Biomedical Engineering*, 14(3):187–192.

[Jaffe et al., 2012] Jaffe, D. L., Nelson, D., and Thiemer, J. (2012). Perspectives in assistive technology. Technical report.

[Kala et al., 2010] Kala, R., Shukla, A., and Tiwari, R. (2010). Fusion of probabilistic a* algorithm and fuzzy inference system for robotic path planning. *Artificial Intelligence Review.*

[Lopes et al., 2013] Lopes, A., Pires, G., and Nunes, U. (2013). Nunes assisted navigation for a brain-actuated intelligent wheelchair. *Robotics and Autonomous Systems*, (245-258).

[Lopes et al., 2007] Lopes, A. C., Moita, F., Nunes, U., and Solea, R. (2007). An outdoor guidepath navigation system for amrs based on robust detection of magnetic markers. *12th IEEE Conference on Emerging Technologies and Factory Automation.*

[Lopes, 2012] Lopes, A. C. B. P. (2012). *Mobile Robot Assisted Navigation based on Collaborative Control.* PhD thesis, University of Coimbra, Faculty of Science and Technology, Department of Electrical and Computer Engineering.

[Marder-Eppstein, 2015] Marder-Eppstein, E. [online]. (2015). Available from: `http://wiki.ros.org/navigation`.

[Miller, 2006] Miller, D. P. (2006). Assistive robotics: An overview. *Assistive Technology and Artificial Intelligence.*

[Park et al., 2012] Park, J., Johnson, C., and Kuipers, B. (2012). Robot navigation with model predictive equilibrium point control. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS).*

[Patel et al., 2012] Patel, M., Miro, J. V., and Dissanayake, G. (2012). Probabilistic activity models to support activities of daily living for wheelchair users. *IROS 2012 Workshop on Progress, challenges and future perspectives in navigation and manipulation assistance for robotic wheelchairs.*

[Perdigão, 2014] Perdigão, J. (2014). Collaborative-control based navigation of mobile human-centered robots. Master's thesis, University of Coimbra.

[Pires and Nunes, 2002] Pires, G. and Nunes, U. (2002). A wheelchair steered through voice commands and assisted by a reactive fuzzy-logic controller. *Intelligent and Robotic Systems*, (34):301–314.

[Poorten et al., 2012] Poorten, E. V., Demeester, E., Reekmans, E., Huntemann, A., and Schutter, J. D. (2012). Powered wheelchair navigation assistance through kinematically correct environmental haptic feedback. *Robotics and Automation (ICRA), 2012 IEEE International Conference on.*

[Schnabel et al., 2007] Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient ransac for point-cloud shape detection. *Computer Graphics Forum.*

[Sousa et al., 2007] Sousa, P., Araújo, R., Nunes, U., Alves, L., and Lopes, A. (2007). Real-time architecture for mobile assistant robots. *12th IEEE Conference on Emerging Technologies and Factory Automation.*

[Tsalicoglou and Perrin, 2009] Tsalicoglou, C. and Perrin, X. (2009). Survey on navigation assistants for people with disabilities. Technical report, Autonomous Systems Laboratory ETHZ.

[Tsu-Zen Hong, 2013] Tsu-Zen Hong, Hsin-Han Chiang, Y.-L. C. (2013). Design and implementation of human-robot collaborative control for wheeled mobile robots. *2013 CACS International Automatic Control Conference.*

[Urbano et al., 2009] Urbano, M., Fonseca, J., Nunes, U., and Lopes, A. (2009). A new approach for operating powered wheelchairs by people with severe impairments.

[Vanhooydonck et al., 2010] Vanhooydonck, D., Demeester, E., Hantemann, A., Philips, J., Vanacker, G., Brussel, H. V., and Nuttin, M. (2010). Adaptable navigational assistance for intelligent wheelchairs by means of an implicit personalized user model. *Robotics and Autonomous Systems*, 58(8):963–977.

[Welch and Bishop, 2006] Welch, G. and Bishop, G. (2006). An introduction to the kalman filter.

[Yanco, 1998] Yanco, H. A. (1998). Wheelesley, a robotic wheelchair system: Indoor navigation and user interface. *Assistive technology and artificial intelligence.*

[Yand and Forstner, 2010] Yand, M. and Forstner, W. (2010). Plane detection in point cloud data. Technical report, Department of Photogrammetry Institute of Geodesy and Geoinformation University of Bonn.