DESIGN OF HAPTICALLY ENABLED WHEELCHAIR FOR ASSISTIVE AUTONOMY

by

ARJUN MANI GUPTA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2018

To my parents

Acknowledgements

I would like to express my deepest gratitude to Dr. Gergely Záruba and Dr. Manfred Huber for giving me this opportunity and giving me valuable advice not limited to academics. I would like to thank Dr. Christopher D McMurrough for guiding me and giving constant support. I would also like to extend thanks to The University of Texas at Arlington's Computer Science and Engineering Department for making this possible.

Last but not the least I would like to thank my friends and family for believing in me and supporting me in every way possible through this journey.

April 12, 2018

Abstract

DESIGN OF HAPTICALLY ENABLED WHEELCHAIR

FOR ASSISTIVE AUTONOMY

Arjun Mani Gupta, MS

The University of Texas at Arlington, 2018

Supervising Professors: Gergely Záruba and Manfred Huber

The first records of wheeled seats being used for transporting disabled people date to 8th century in China, however the wheelchair has evolved tremendously since its inception. An electric-powered wheelchair, commonly called a "powerchair" is a wheelchair which incorporates batteries and electric motors into the frame, and so it can be controlled by either the user or an attendant. This control is most commonly done via a small joystick mounted on the armrest, or on the upper rear of the frame. For users who cannot manage a manual joystick, head-switches, chin-operated joysticks, sip-and-puff controllers or other custom controls may allow independent operation of the wheelchair. Although these interfaces make the wheelchair easier to operate, they do not help the user to navigate, nor do they make the user aware of the obstacles in the path of the wheelchair.

In this thesis we are exploring how haptic feedback can be coupled with a short-range navigation system to make powerchairs smarter. More precisely, we employ and modify an off-the shelf haptic joystick for both getting the intended direction of motion from the user as well as providing the user with the haptic feedback in case there are obstacles that prevent the powerchair to move on its current trajectory. The smart powerchair builds and maintains a map based on obstacles that are sensed with a LIDAR (Light Detection and Ranging) using simultaneous localization and mapping (SLAM); this information is then used to provide haptic feedback and perform navigation and obstacle avoidance. The

system contains custom designed hard- and firm-ware to communicate with the original wheelchair controller. The overarching custom software architecture is designed and built over ROS (The Robot Operating System).

TABLE OF CONTENTS

List of Illustrations

List of Tables

Chapter 1

INTRODUCTION

With the available technological advancements today and the trend where most of the tasks are being automated, it is natural to get lost in the urge of making things completely autonomous even where such changes are mostly unwelcome by the target user population of that particular device. The objective is definitely honorable but in the process of getting to the cutting edge the soul of the problem is lost which is to assist people with disabilities.

Initial work in the field of “assisted wheelchairs” has been revolving around the idea of making the wheelchair an autonomous unit. A plethora of sensors have been used with the aim of enabling the wheelchair to autonomously navigate to a given goal in an environment. These sensor systems have even been put together to form advanced control systems, enabling the user to maneuver the wheelchair through movements of the chin, or other gestures.

Most of these seem to be targeting a population with severe motor disabilities. However, the majority of the wheelchair users are not affected to such an extent that they need complete autonomy as seen in the info graphics shown in figure 1-1. For this type of users a semi-autonomous system would likely be a better fit. The built in intelligence of the wheelchair must communicate to the user what it sees or what action it is about to take, rather than just executing it, thereby assisting the user and not substituting them. Possible methods of communication are through display and sound (Toyota’s I-Real Personal Mobility Concept, debuted in 2007, uses sound to indicate that the user is too close to an obstacle), with haptics being another option. In this thesis, we are exploring how haptic feedback can make powerchairs more intuitive to use; to make the chair even smarter, we will use a short-range navigation system as well.

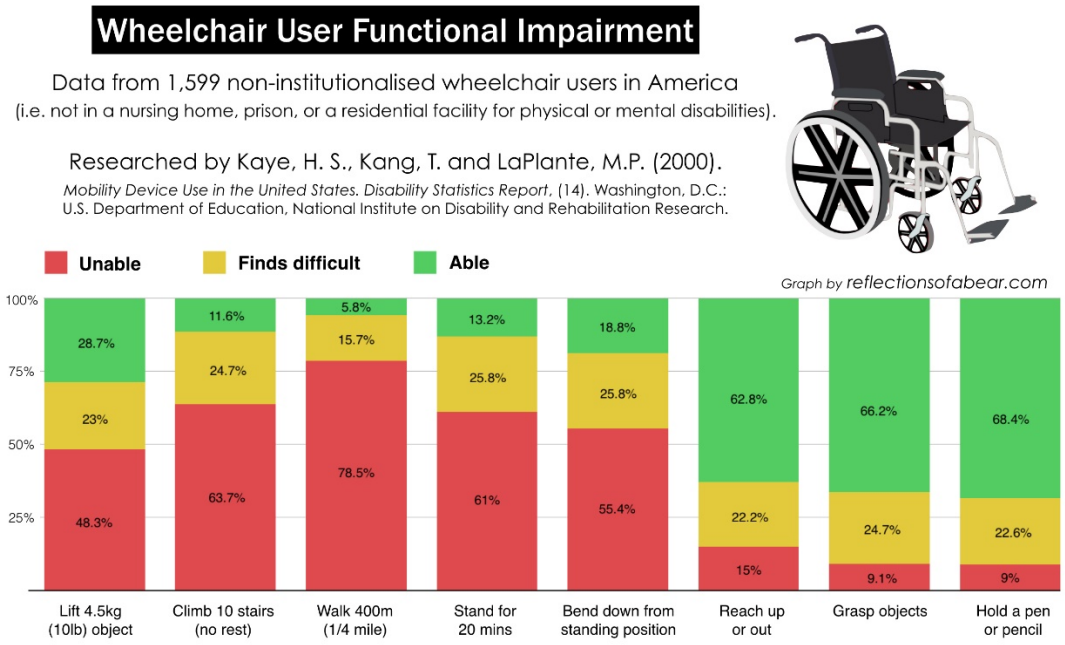

Figure 1-1 Wheelchair User Functional Impairment [1]

Haptics was selected as the mode of communication with the user in this project because of it is an intuitive feedback methodology that does not distract the user (as opposed to visual or auditory stimuli) and provides a more natural integration with the short-range navigation system (semi-autonomous control).

Haptics can be defined as anything related to or based on the sense of touch. Haptics has proved to be effective in many fields such as tele-surgical robots, bio-manipulation in virtual reality, and remote operation of various unmanned vehicle systems [2, 3, 4, 5]. There have also been attempts to apply haptics to wheelchairs with degree of success [6, 7, 8]. The core foci of these works have been on perfect obstacle avoidance. On the other hand, the questions we are trying to answer are:

- How to balance autonomy and user input to achieve an assistive outcome?

- How to infer the intended direction of motion from the user and eventually guide the user to his/her goal?
- How to make the user aware of the obstacles around wheelchair?

Finding a solution for these questions has been the motivation behind this thesis, taking a synthesis route where we design and implement a semi intelligent system that helps the user to navigate using haptics to keep the user informed about the obstacles around the wheelchair; this not only helps in avoiding obstacles but also enables the user to keep an healthy distance between the wheelchair and any obstacles. The framework conceived herein is semi-autonomous; it makes sure the control is in the hands of the user while allowing the sensors to feedback environmental information to the user.

Our system has three major components:

1. Obtaining the intended direction from the user and translating it to a direction of motion for the system to travel;
2. Obstacle avoidance to avoid and maintain a safe distance from all the obstacles around the wheelchair within a certain radius;
3. Obtaining these obstacle information and turning them to haptic feedback using the force-feedback system to provide feedback to the user.

The rest of this thesis document is organized in the following way: Chapter 2 describes background work, evolution of the wheelchair and some overview of the important concepts like haptics, force-feedback and the ROS (Robotic Operating System) which are necessary for understanding the design and implementation details contained in this thesis. Chapter 3 contains the design overview of the project. Chapters 4 and 5 detail the implementation. Chapter 6 concludes the thesis and discusses possible improvements and future work.

Chapter 2

BACKGROUND

2.1 Wheelchair History

The first records of wheeled seats being used for transporting people with disabilities date to 8th century in China; however the wheelchair has evolved tremendously since its inception. Some of the different types of wheelchairs are:

- Manual self-propelled wheelchairs
- Manual attendant-propelled wheelchairs
- Powered wheelchairs
- Mobility scooters
- Standing wheelchairs
- All-terrain wheelchairs
- Self-balancing wheelchairs
- Sports wheelchairs

An electric-powered wheelchair, commonly called a "powerchair" is a wheelchair, which incorporates batteries and electric motors into the frame, thus it can be controlled by either the user or an attendant. This control is most commonly done via a small joystick mounted on the armrest, or on the upper rear of the frame. For users who cannot manage a manual joystick, head-switches, chin-operated joysticks, sip-and-puff controllers or other custom controls may allow independent operation of the wheelchair [9].

2.2 Smart Wheelchairs

Smart wheelchairs have been the subject of research since the early 1980s and have been developed all over the world [10]. Table 2-1 lists some of the attempts that

were made in the smart wheelchair field. These advancements were motivated by the wheelchair user group, i.e., people who have visual field reduction, low vision, tremors, cognitive deficiencies or spasticity. People with such disabilities often lack the capability to move independently and rely on a chaperone to move them around with a manual wheelchair. In general, most of the developments in the filed have been either towards obstacle avoidance or force feedback.

Smart Wheelchair	Description
Automated-Guided Wheelchair NEC Corporation, Japan	Follows tracks laid out with magnetic ferrite marker tape. Uses IR sensors to stop when obstacles detected in its path.
Autonomous Wheelchair Arizona State University, U.S.	Uses machine vision to identify landmarks and center wheelchair in hallway.
CHARHM CDTA, Algeria	Chair navigates autonomously to location in environment based on internal map and information from machine vision.
COACH French Atomic Energy Commission, France	Provides obstacle avoidance and follows walls. Unclear how active operating mode is chosen.
National University of Singapore, Singapore	Uses dead reckoning to keep wheelchair on prescribed path. User can leave path to avoid obstacles, and controls speed of wheelchair. Path can be defined with GUI or by walkthrough.

Table 2-1 Smart wheelchairs reported in literature [10].

2.3 Haptics Enabled Wheelchair

There have been a few endeavors to make a haptics enabled wheelchair; one such attempt was “Obstacle avoidance using haptics and a laser rangefinder” [11]. In this project the researchers have used a laser rangefinder and a Falcon joystick for sensing and human interface respectively. The laser rangefinder information was used to create a virtual obstacle surface which can be “felt” using the Falcon. This enabled the user to feel her surrounding while exploring the environment. Indeed, the target population for that project was people with visual impairments.

There are projects other projects that aim to interact with the wheelchair, e.g., via voice command, eye tracking, Myoelectrodes etc. There have also been systems proposed and designed with cameras to detect gestures [12] or the user’s head position to interact with the wheelchair [13]. In one of the more recent development efforts EEG is used to create a “brain controlled” wheelchair [14].

2.4 Haptics

Haptics is anything related to or based on the sense of touch; haptics applied to an I/O device is often referred to as *force-feedback*. Just like any other computer interface device, a “force-feedback” device is meant to give input and/or output capabilities to the end user. For example, a speaker can interface with a user by providing output in the form of sound; a microphone interfaces with users by receiving the input in the form of sound; displays on the other hand provide output in the form of light to give us visual information. Similarly, a force-feedback device provides output in the form of haptics to convey the information in the form of touch sensations to the user. The touch sensations that it can provide ranges from simple feeling of vibrations to complex effects such a feeling friction, appearing as if moving through liquid, simulating the feel for the density of a material, motion, g-forces and many more.

2.5 Force-feedback devices

Haptics have been used in plethora of touch-based interfacing devices like flight yoke joysticks, steering wheel control for games, gamepads, mice and pointing devices, e.g., the Novint Falcon as seen in figure 2-1. Of the listed devices the most tested and widely used force-feedback device is a joystick. Simply described a joystick is a vertical post with a grip on it pivoted at one end which can move freely about the pivot in a 2D Cartesian plane. These movements are recorded using sensors and used for processing. As most wheelchair users are already accustomed to joysticks, we decided to use a force-feedback joystick to interact with the user.

Figure 2-1 Various Force-feedback devices. Top left: Mouse, Right: Steering control; Bottom left: Novint Falcon, Right: Microsoft Force-feedback joystick

2.5.1 Force-feedback Joystick Construction

The three major components of a force-feedback joystick are the actuators, gimbal and power transfer mechanism. Apart from these the joystick can be as diverse as possible in terms of shape, size and number of buttons. Some of them may even have a throttle knob. The power transmission connects the motor to the gimbal which in turn connects to the actual control stick. One of the widely used mechanism for the gimbal is the double slotted bale [15] as seen in figure 2-2. The double slotted bale's mechanics consists of orthogonally placed slotted curves via which the control stick slides. The movement of each of the slotted curves is detected by an encoder to retrieve the position of the control stick.

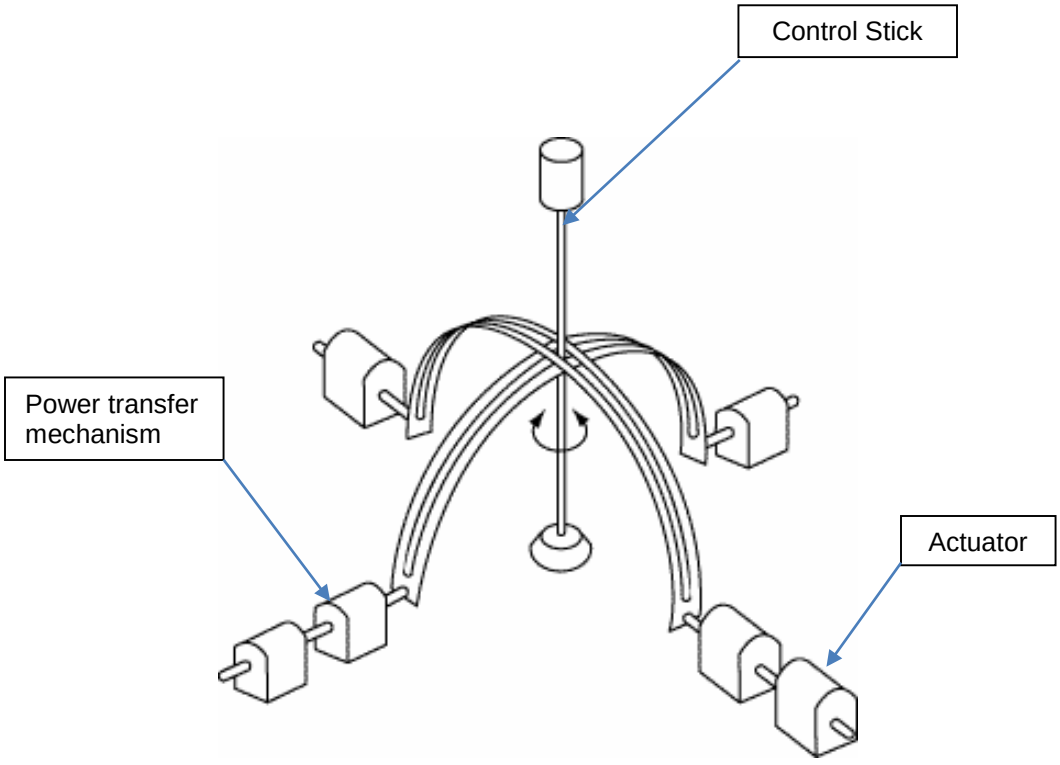

Figure 2-2 Double Slotted Bale [15]

The power transfer mechanism bridges the gimbal and the actuator by a cable or belt or a gear train system, therefore giving the users the feel that the motors can provide. These mechanisms are designed in such a way that they are back drivable. The actuators are the primary source of force in the force-feedback system. Depending on the motors' capabilities the range of the force experienced during force-feedback may vary.

2.5.2 Force-feedback effects

In order to activate force-feedback the attached computer sends data packets to the joystick. The complete data packet sent to the force-feedback device with the information on which haptic to play is called an *effect*. These *effects* can be classified into six categories:

1. ***Static*** as the name suggests is fixed; static effects can be created and sent to the joystick prior to the time they need to be presented to the user. They can be constructed manually or we can choose the inbuilt effects to be played as static effects. The device can hold a number of effects proportional to RAM on board.
2. ***One-shot*** effects play for a set duration of time and terminate once done.
3. ***Time-based*** effects are the effects are periodic function representation of effects like sinusoidal motion etc.
4. ***Dynamic*** effects can be changed as they are being played for example with respect to the position of the joystick or any other condition for that matter; they are very complicated to design and test but they do provide much better tactile sensation to the user.
5. ***Open-ended*** effects unlike one-shot effects do not have a termination time set, they plays until explicitly canceled by the system. This also

implies that the system has to monitor the effect state at every instant so that it can stop the effect at the appropriate time.

6. ***Interactive*** effects depend on the joystick state. Some of the interactive effects include simulation of a spring, friction etc.

For example, the recoil of a gun could be represented with one-shot, static and time-based effects whereas driving a speed boat would need interactive, dynamic and open ended effects.

2.6 ROS

The following excerpt is quoted from [16]: “The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

Why? Because creating truly robust, general-purpose robot software is hard. From the robot's perspective, problems that seem trivial to humans often vary wildly between instances of tasks and environments. Dealing with these variations is so hard that no single individual, laboratory, or institution can hope to do it on their own.

As a result, ROS was built from the ground up to encourage collaborative robotics software development. For example, one laboratory might have experts in mapping indoor environments, and could contribute a world-class system for producing maps. Another group might have experts at using maps to navigate, and yet another group might have discovered a computer vision approach that works well for recognizing small objects in clutter. ROS was designed specifically for groups like these to collaborate and build upon each other's work.”

Figure 2-3 ROS Equation [16]

ROS is a comprehensive system with most of the necessary capability, tools, network and support system as seen from the “ROS Equation” depicted in figure 2-2. It is completely open source hence enabling the creation of add on the features that one feels are missing for the application for which they intend it to use ROS. The ROS ecosystem now consists of tens of thousands of users’ worldwide, working in domains ranging from tabletop hobby projects to large industrial automation systems [16].

2.6.1 *Core Components*

ROS ecosystem has grown exponentially to an extent that providing a comprehensive list of all components in this thesis is impractical but the core components of the system deserve some explanation. Broadly the core components include the communication infrastructure, robot specific features, and tools. We will detail them in the following sections.

2.6.1.1 *Communications infrastructure*

ROS middleware mainly revolves around the idea of a publish/subscribe mechanism. The communication infrastructure provides features like publish/subscribe anonymous message passing, recording and playback of messages, request/response remote procedure calls and a distributed parameter system.

Communication is analogous to our spinal cord when it comes to a robot application. ROS’s built in communication infrastructure is a well-tested messaging

system that enables us to implement a new robot application without worrying about the management of communication between distributed nodes through the anonymous publish/subscribe mechanism. This also implicitly drives the implementation of a well-structured interface between the nodes in the system which in turn improves code reusability and encapsulation.

The ROS communication feature provides us with the capabilities of recording and playback of messages; as the system uses anonymous publish/subscribe mechanism the data can be easily recorded and played back without any major changes to the underlying code. This is a very powerful design pattern as it ensures modularity of the system. Inter-process communication is also supported by this infrastructure.

A distributed parameter system also provides the provision to describe various global parameters that define the system which later be accessed by any other node for reference or change the configuration of other tasks.

2.6.1.2 Robot-Specific Features

Some of the common capabilities provided by the ROS ecosystem libraries and tools enabling quick deployment of various robots are the: Standard Message Definitions for Robots, Robot Geometry Library, Robot Description Language, Preemptable Remote Procedure Calls, Diagnostics, Pose Estimation, Localization, Mapping and Navigation

The Standard Message Definitions for Robots is a set of predefined message structures that can be used by the developer while implementation to ensure seamless communication between all components within the ROS ecosystem. Examples of these messages are geometric concepts like poses, transforms, and vectors; for sensors like cameras, IMUs and lasers; and for navigation data like odometry, paths, and maps; among many others [17].

Robot Geometry Library makes it easier to track where each part of the robot is by using transformations between them. This is used to manage the coordinate transformation data of the robot and it is designed with such efficiency that it can handle more than hundred degrees of freedom with update rates of hundreds of Hertz. This is supported by the Robot Description Language which provides a way to describe the robot which is being worked on in a machine-readable way.

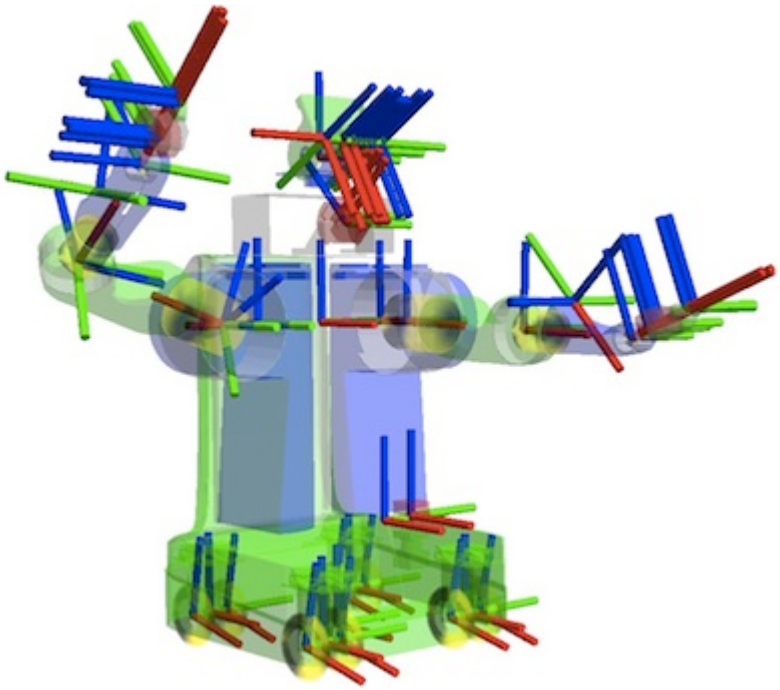

Figure 2-4 Robots frames in a PR2 robot [17]

Preemptable Remote Procedure Calls help in keeping track of the state of the robot when a task is being executed. Actions are used in ROS for this purpose. ROS also provides a diagnostics service which provides the aggregated status of all the robot components at a glance.

Pose Estimation, Localization, Mapping and Navigation are some of the inbuilt features that ROS ecosystem provides for rapid prototyping (figure 2-4) which help to solve some the basic robotics problems stated above.

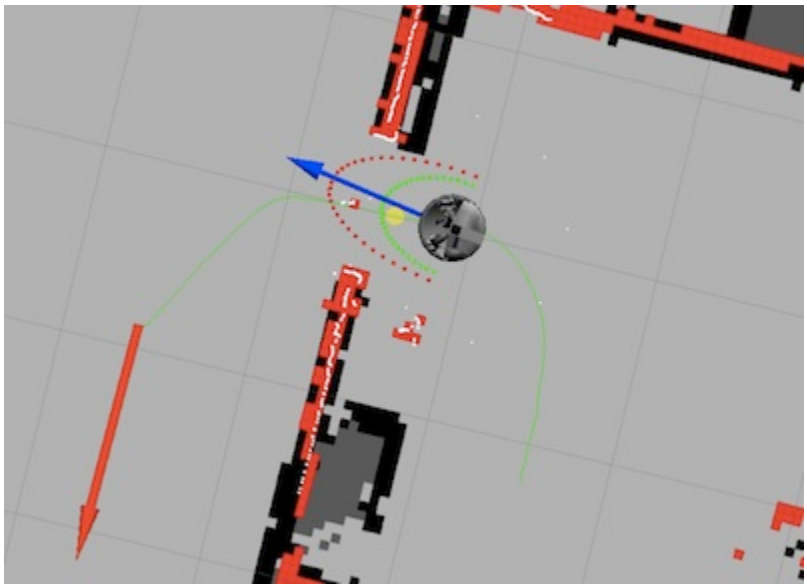

Figure 2-5 ROS Pose Estimation, Localization, Mapping and Navigation [17]

2.6.1.3 *Tools*

One of the well praised and strongest feature of ROS is the robust development suite that it comes with. The tools help in debugging, introspecting, plotting and visualizing the state of the robot system under development. ROS provides a native non-GUI (45+) command line tool interface at our disposal.

Rviz is one of the most popular general purpose, three-dimensional visualization tool used for many sensor types and any robot described by the robot description language (as we have seen in figure 2-5). Rviz can visualize many of the common message types provided in ROS, such as laser scans, three-dimensional point clouds, and camera images. It also uses information from the tf library to show all of the sensor data in a common coordinate frame, together with a three-dimensional rendering of the

robot. Visualizing all data in the same application not only looks impressive, but also allows a quick view to what the robot sees, and thus help identify problems such as sensor misalignments or robot model inaccuracies[17].

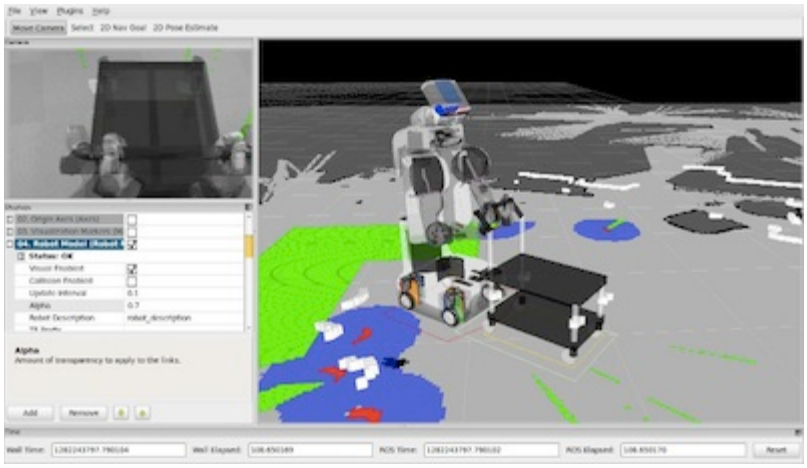

Figure 2-6 rviz sample screenshot of PR2 looking at the table in front [17]

Another powerful ROS tool is rqt which is a Qt-based framework for developing graphical user interfaces for the robot. There are multiple rqt plugins which enable us to visualize various aspects of the system, e.g., rqt_graph shows how each of the robot nodes are connected to each other, rqt_plot helps to monitor any data that we wish to plot for observation, rqt_topic and rqt publisher plugins can be used to monitor the communication activity that's going on the system, rqt_bag can be used for data logging and playback. Developers can also develop custom rqt plugins to further enhance ROS' capabilities.

Chapter 3

DESIGN OVERVIEW

The goal for the assistive autonomy model imagined in this thesis for a haptically enabled intelligent wheelchair is focused on two main ideas: one is the ability to incorporate the manual control with the on board short range navigation system and the other is to make the user aware of the obstacles around the wheelchair. These ideas can be clearly seen reflected by the design of the system. The system has two interconnected components one is the short range navigation system and the other is the haptic feedback system as seen in figure 3-1. The former takes care of detecting the obstacles in the environment around the wheelchair and how to navigate around them as per the user instruction and the later conveys where the obstacles are to the user via haptics.

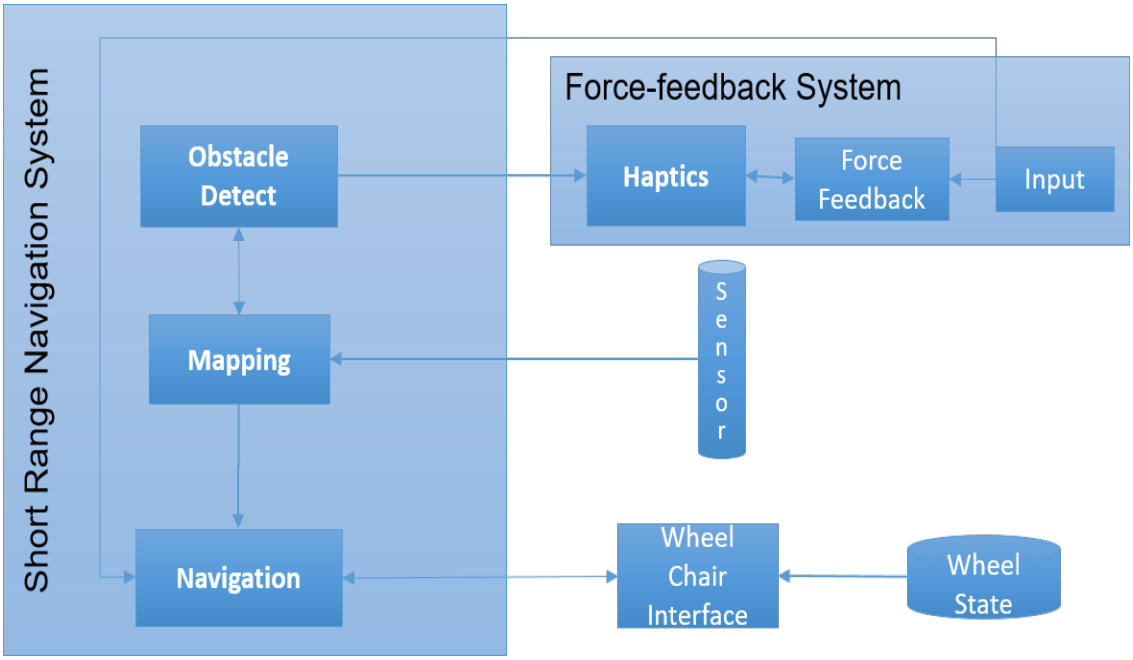

Figure 3-1 Design diagram

3.1 Short Range Navigation System

This system takes in the user input which gives us the intended direction of travel; it then combines this information with the detected obstacles and the map to find a clear path in the commanded heading. The major sub-systems that make up this system are mapping, obstacle detection and navigation.

3.1.1 *Mapping*

Occupancy grids were first popularized by Hans Moravec and Alberto Elfes at CMU. An occupancy grid as seen in figure 3-2 is a two dimensional grid structure in which each grid cell can be unknown, free-space or an obstacle. If the state of a cell is unknown

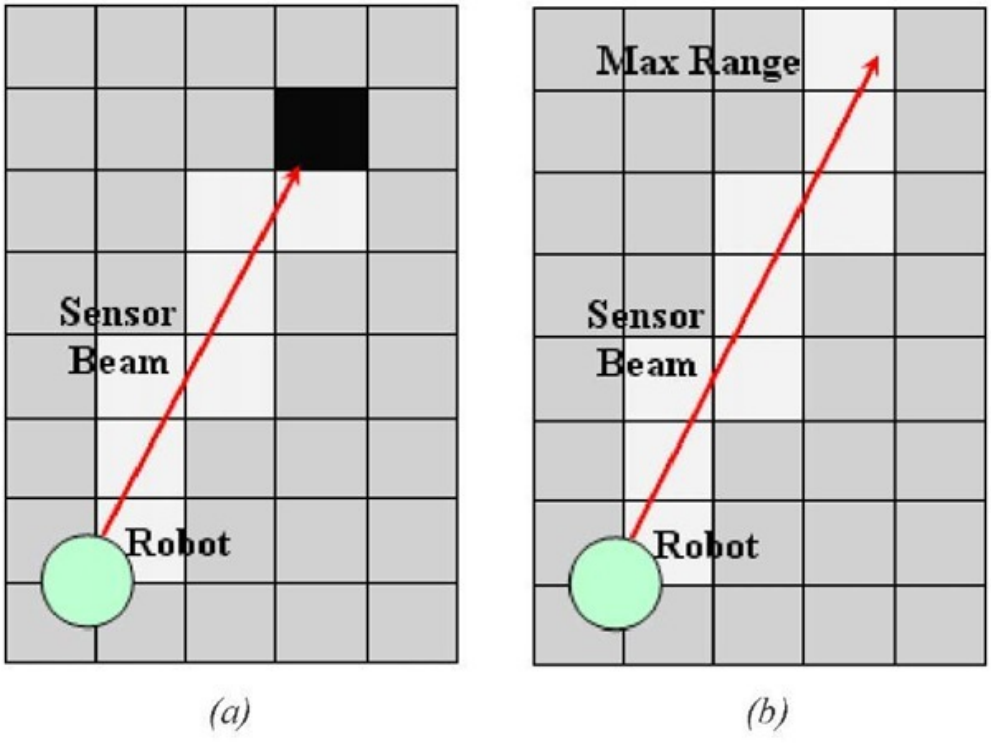

Figure 3-2 (a) Area with an obstacle; (b) area without an obstacle

then the state of the cell is determined probabilistically. This probability is computed from the data obtained from the sensor readings using Bayes law as shown below,

$$p(A \mid B) = \frac{p(B \mid A) * p(A)}{p(B)}$$

Where A is occupancy (i,j) and B is an observation (i.e. Sensor reading)

Once we have the map using the occupancy grid technique we must know where we are in the map this is known as localization. Simultaneous Localization and Mapping (SLAM) is a technique where a map is being built as the agent is localizing itself in an environment. As it can be seen in figure 3-3 A SLAM system can map the environment as the agent is moving through it while localizing the agent in the generated map.

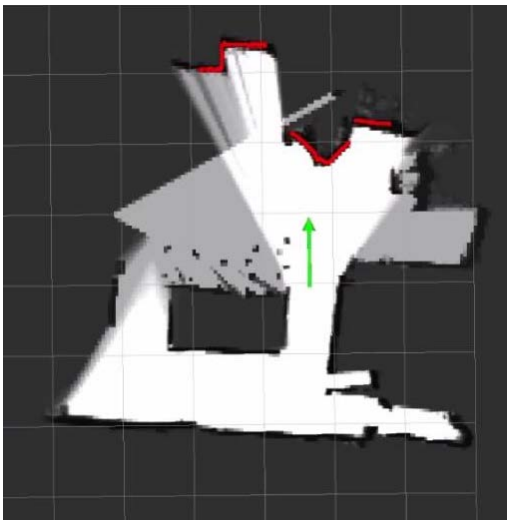

Figure 3-3 Sample graphical representation of SLAM

3.1.2 *Obstacle detection and navigation*

Once the mapping is done we can detect the obstacles from the occupancy grid with the state information of each cell. This information is then used to perform navigation using path planning. One the most popular path planning algorithms (and the one that we are going to use here) is the Manhattan distance path planning algorithm.

The distance between two points in a grid based on a strictly horizontal and/or vertical path (that is, along the grid lines), the Manhattan distance is the simple sum of the horizontal and vertical components as seen in figure 3-4. This is done with the grid structure generated by SLAM in the intended direction of travel avoiding the obstacle cells and the buffer cells (helps the robot to maintain safe distance from the obstacle) to find a clear path of travel in the intended heading.

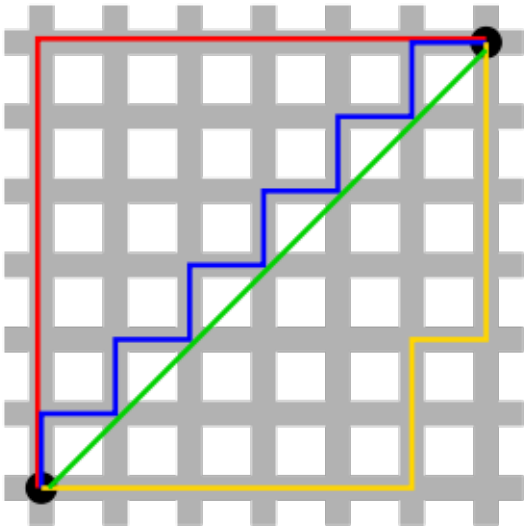

Figure 3-4 Green: diagonal, straight-line distance. Red, Blue, yellow: equivalent Manhattan distances. [18]

3.2 Force-Feedback system

Force-feedback system consists of haptics and the force-feedback device. The haptic feedback provided is derived based on the generated map and obstacles. We investigate at a defined radius around the system to determine where the obstacles are while checking what position the joystick is in and then deciding what haptic to provide to the user. This decision is made by the haptics module. For example, if the user tells the system to go forward but the system sees an obstacle in front of the powerchair then the force-feedback device pushes back the user's hand and offers resistance in the direction

of the obstacle, hence letting the user know about the obstacle in a non-intrusive and intuitive way.

Chapter 4

IMPLEMENTATION

4.1 Setup

The design discussed in the previous chapter has been implemented using ROS; in addition custom hardware was built to interface with an off-the-shelf power wheelchair. As can be seen in figure 4-1, ROS runs on an Intel NUC which handles the mapping, navigation, obstacle detection, control & odometry, haptics node and the communication node. The force-feedback device we employed is a Microsoft Force-feedback 2 joystick which is modeled after a flight yoke. The sensor used for scanning the environment is a Hokuyo UTM-30LX LIDAR (light detection and ranging) device.

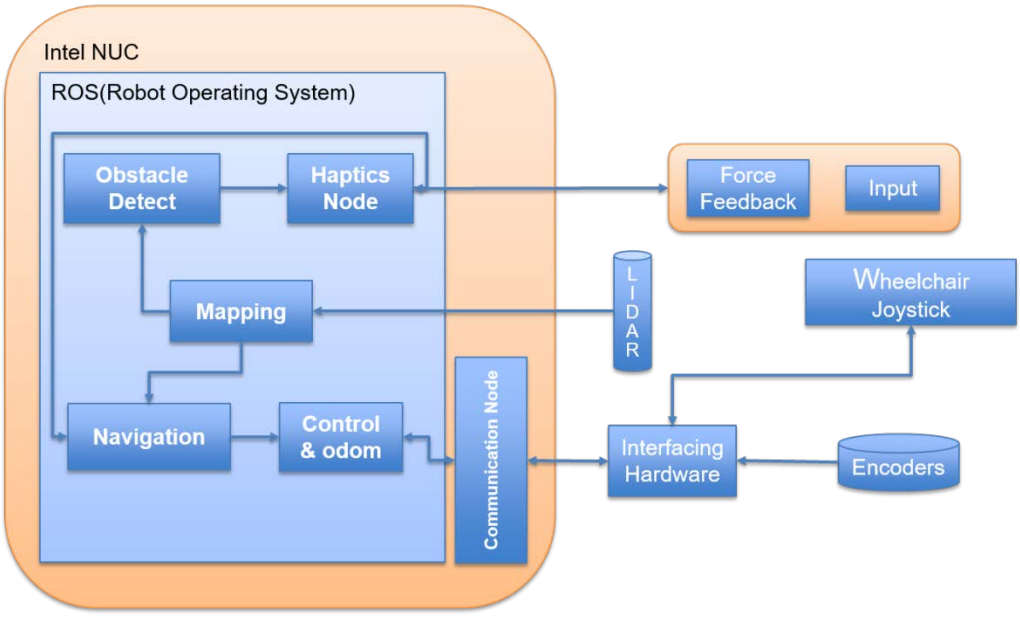

Figure 4-1 Implementation setup

4.2 System Overview

As it can be seen in figure 4-1, the LIDAR scans the environment and streams the data to the Intel NUC. There, the mapping component receives such data and performs SLAM. The resulting map and localization data is then used by both navigation and obstacle detection system to perform their respective tasks. The navigation with the resulting info from SLAM and the user input is then used to provide navigation commands to the control and odometry subsystem which in turn computes the necessary effort required to get the wheelchair to the commanded state with the information received from the encoders via the interfacing hardware and the communication node. The computed commands are then sent to the wheelchair through the communication node which delivers it to the interfacing hardware which translates the commands to voltages to actuate the wheelchair. The obstacle detection system determines from the SLAM information where each obstacle is in a defined radius and conveys this information to the haptics node which then generates the force-feedback commands and passes it down to the joystick to be executed. Next we will look at each of the sub components in detail.

4.3 Interfacing Hardware

The basic idea of this piece of hardware is to enable control of the wheelchair using a computer and also to receive the encoder reading and relay it back to the

computer. This can be seen in the overall schematic as seen in figure 4-2. We can observe that there is a component

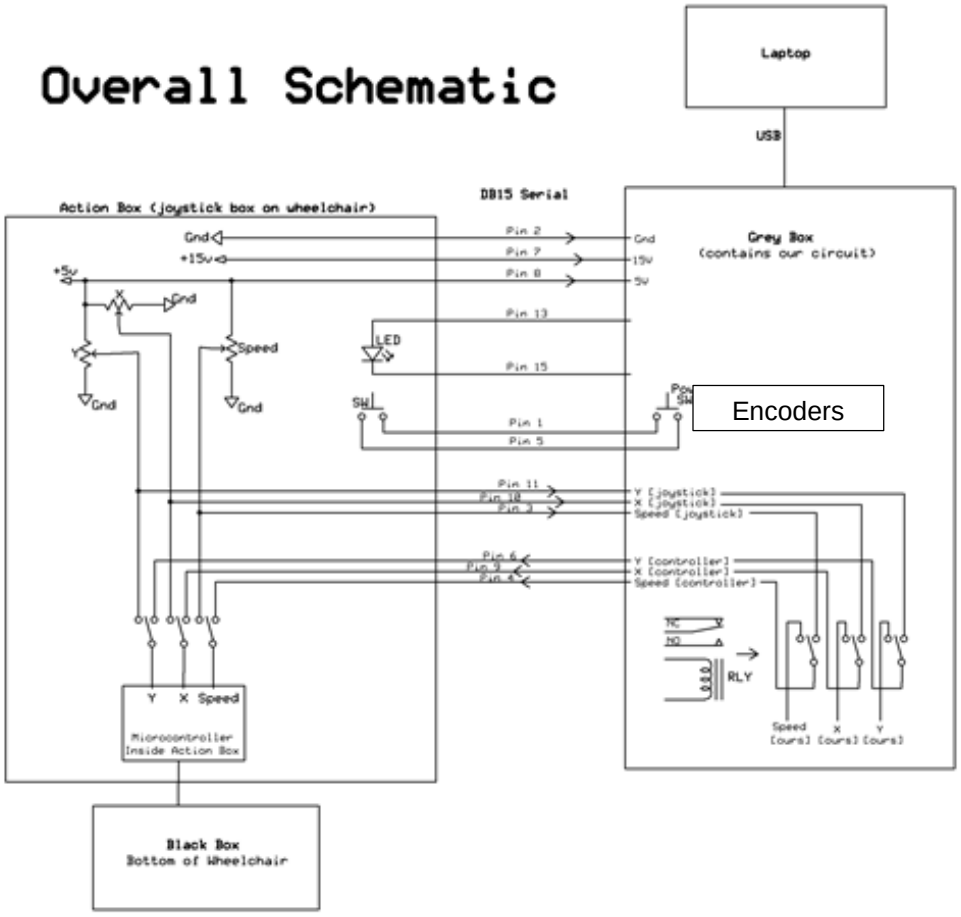

Figure 4-2 Overall Schematic

labeled the *black box*; this is the motor controller of the wheelchair with which it came. It receives the control instructions from the wheelchair joystick labeled as the *action box*. We redirected the connection between the original joystick and the action box by injecting a switchable digital potentiometer. The section labeled

grey box contains all the custom hardware that helps in serially communicating with the computer via USB and also getting the encoder readings.

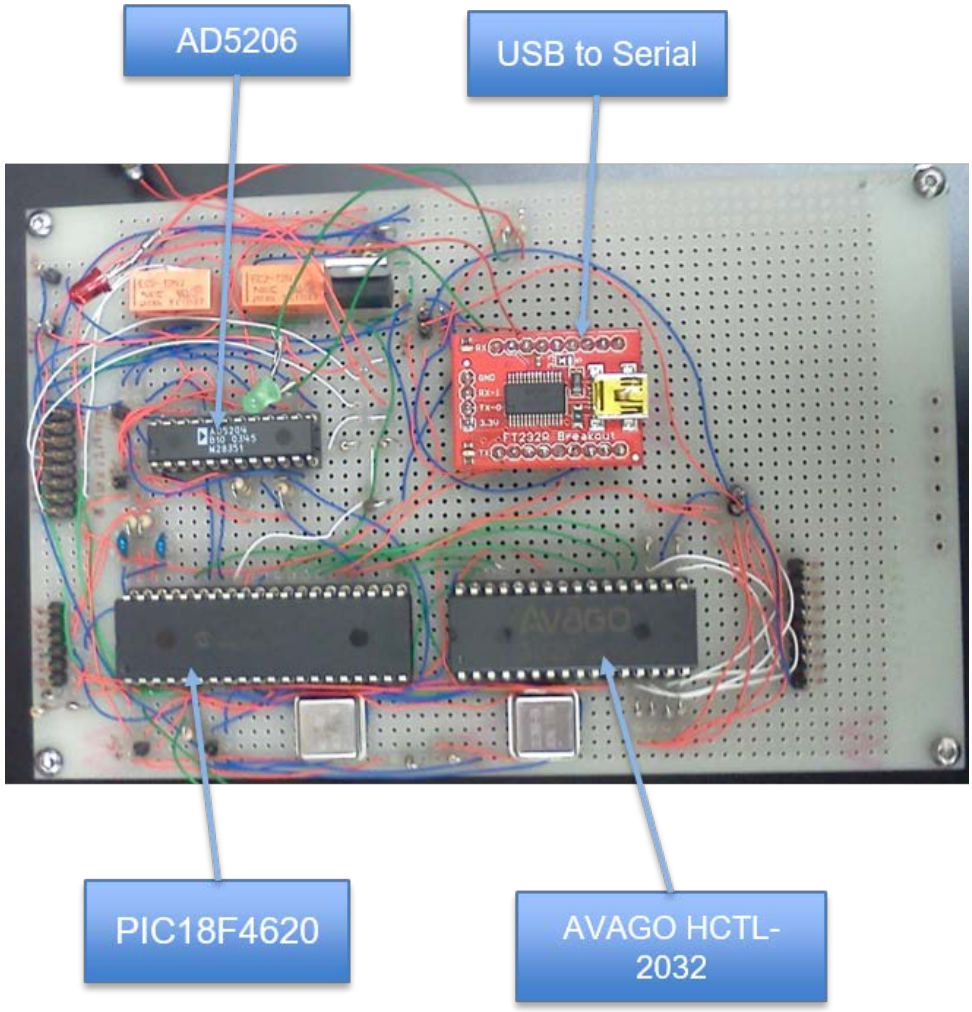

Figure 4-3 Interfacing board

As we can see from figure 4-3 we are using a PIC18F4620 as our microcontroller which was programmed using MPLAB X IDE v3.50. The AVAGO HCTL-2032 is an encoder counter with 2 channels also has a 33MHz clock speed. The encoder counter keeps track of the encoder counts and provides

it to the microcontroller when polled. The AD5206 is the digital potentiometer we are using to control the joystick of the wheelchair. We use an FT232R as a USB to Serial converter. All of this is placed inside an enclosure as seen in figure 4-4 with the appropriate ports for ease of access. There is also an emergency control toggle switch on both the enclosure box as well as the wheelchair joystick.

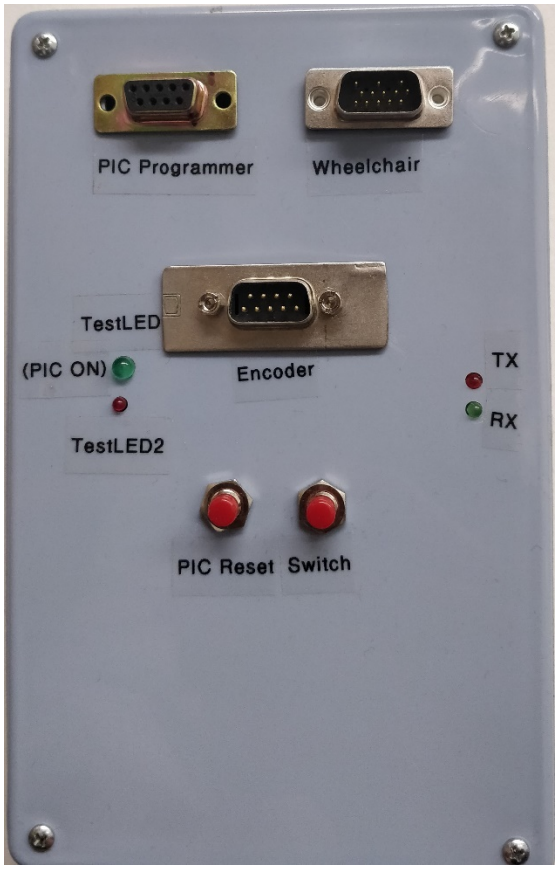

Figure 4-4 Enclosure box

For the computer to talk with the interfacing hardware a middleware was designed in C++ and later integrated with ROS. The communication node takes

care of the publishing and retrieval of the information with the interfacing hardware.

4.4 Mapping, Navigation and control

The integral components of mapping, navigation and control are the ROS systems: hector slam, move base and PID. Let us see each of them in detail.

4.4.1 *Mapping and Localization*

We use a LIDAR for the sensor required for mapping and we are doing SLAM with the help of hector_slam package in ROS. This helps us not only to map the surrounding but also to estimate the current pose of the wheelchair. hector_slam uses the hector_mapping node for learning a map of the environment and simultaneously estimating the platform's 2D pose at laser scanner frame rate. Hector_slam also publishes the map to the map topic which is then subscribed to by the navigation algorithm for processing.

4.4.2 *Navigation*

For navigation we use the move_base package in ROS whose structure is

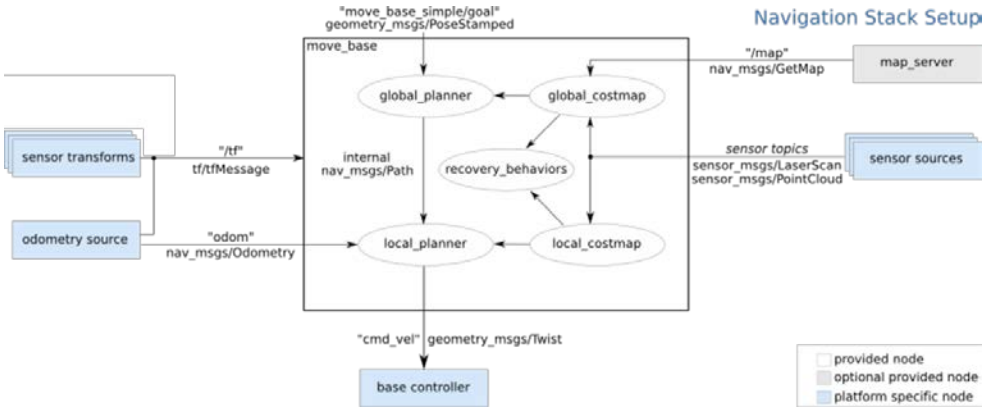

Figure 4-5 Move base system overview [19]

depicted in figure 4-5. The map_server here is the map provided by the hectore_slam and base controller is the control and odometry component of our system. The sensor source is the LIDAR. The map server consists of two cost maps one local and the other global. This helps to greatly improve the efficiency in planning and executing a navigation task as the global cost maps focuses on getting the overall path planning done while the local cost map figures how to avoid immediate obstacles. Move base also has an inbuilt recovery mechanism which helps the robot to get out of situations in which it gets stuck by executing a series of recovery behaviors. The input for the navigation is given as a unit vector in the desired heading obtained from the joystick.

4.4.3 *Control*

The velocity commands from the navigation module have to be translated into X and Y (voltage) values of the wheelchair joystick to get complete control of the system where X controls the linear component of the velocity and Y controls the angular velocity component. To achieve this two PID controllers where used. The PID was implemented using the pid package in ROS with use of suitable of gains. The gain was calculated by following a standard procedure in which first all gains are set to zero then P is increase until the response to a disturbance is steady oscillation. After this D is increased until the oscillations go away (i.e. the system becomes critically damped). The previous two steps are repeated until increasing the D gain does not stop the oscillations. Then the I gain

is increased until the oscillations are reduced to the desired level (normally zero but a quicker response can be achieved with some tolerance of oscillations).

In the ROS PID used the control_effort was set to the joystick values of the wheelchair, the state is the velocity feedback from the encoders and the setpoint is the linear and angular velocity components from the twist messages published by the navigation module. The wheelchair joystick has a built in dead-band which was manually compensated in the control node.

4.5 Obstacle detection

The obstacle detection subscribes to the local costmap generated by the navigation module. The local map covers small area within a given radius. Then we get the wheelchair's pose information and transform it to the map frame. With the new pose information we scan for any obstacle cells and calculate at which angle each obstacle is situated using *a2tan* with the identified coordinates; the list of obstacles and respective angles are then sent to the haptics node.

4.6 Haptics and Force-feedback

The haptics stimulus is calculated in the haptic node and then the force-feedback command is sent to the Microsoft force-feedback 2 joystick. We also get the intended direction of travel from the joystick and thus provide the navigation module a travel direction vector unless direction is changed or there is an obstacle in the path.

4.6.1 *Haptics*

Once the list of obstacles and respective angles are received then haptics module monitors the current position of the joystick. If the module senses that the joystick is being moved the direction where an obstacle is located then it commands to apply a constant force in the opposite direction of the joystick motion. Once this haptic is decided a force-feedback effect packet is built and sent to the device for rendering.

4.6.2 *Force-Feedback*

The force-feedback joystick we use is a Microsoft force-feedback 2 joystick as shown in the figure 4-6. This joystick was chosen as opposed to other force-feedback joysticks is because it has a stronger effect range.

Figure 4-6 Microsoft Force-feedback 2 joystick

The joystick's force feedback plane has been split into eight sections each forty five degrees apart for the ease of rendering and intuitiveness. For example if there is an obstacle identified at 60 degrees then the force-feedback is applied in the opposite direction to the section 45-90 degrees. This intuitively suggests to the user not to move in that specific direction. The granularity of the sections can be increased based on user experience feedback.

4.7 System Power Management

The wheelchair uses a 24 Volt power system. As the force-feedback joystick requires 120VAC mains supply we decided to incorporate an inverter onboard the wheelchair. A 400 Watt inverter powers both the Intel NUC and the Microsoft Force-feedback 2 joystick. A 400W inverter was deemed to be sufficient after taking into account the peak power requirements of the Intel NUC and the joystick.

4.8 Assembly

As seen in figure 4-7, on the current prototype, the LIDAR is mounted in the front of the wheelchair on a protruding poll which goes in-between the legs of the user hence giving a clear view in front of the wheelchair for accurate obstacle detection. The joystick is mounted on an extended boom in front of the right arm rest.

Figure 4-7 Joystick and LIDAR mounting

The inverter, Intel NUC and the control box are positioned behind the wheelchair as shown in figure 4-8.

Figure 4-8 Inverter, Intel NUC and the control box mounting

4.9 Simulation and Testing

Hardware in loop simulation and testing has been used to evaluate the implemented system. Hardware-in-the-loop (HIL) simulation, or HWIL, is a

technique that is used in the development and test of complex real-time embedded systems. HIL simulation provides an effective platform by adding the complexity of the plant under control to the test platform [20]. The whole system has been tested, including the force feedback and motor control with virtual obstacles in gazebo (a graphical simulator in ROS). The wheelchair was place on a raised platform so as to prevent unintended motion while testing. The PID controller was tested separately with various setpoint velocities; one instance of the graphical plot for a setpoint and system state is shown in figure 4-9.

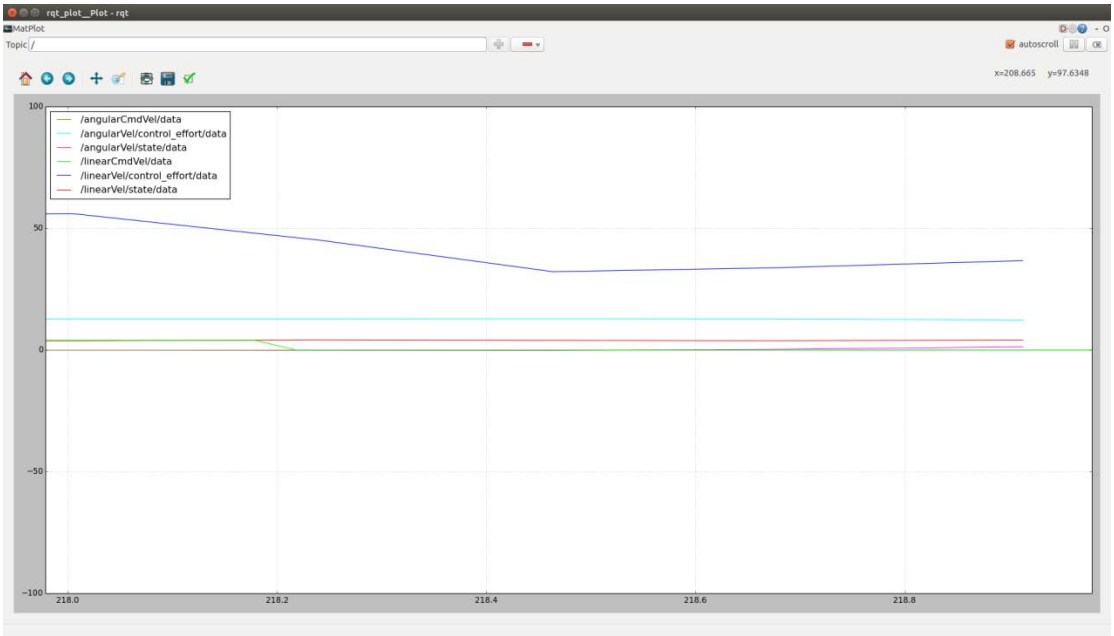

Figure 4-9 Sample rqt_plot of the PID controls and system states

The setpoint was given and the system response was recorded with the data from the wheel encoders. Similarly the virtual obstacles where placed in a simulated world in gazebo and navigation input was given through the joystick.

The force-feedback response and the navigation pattern when approaching virtual obstacles resulted in expected behavior.

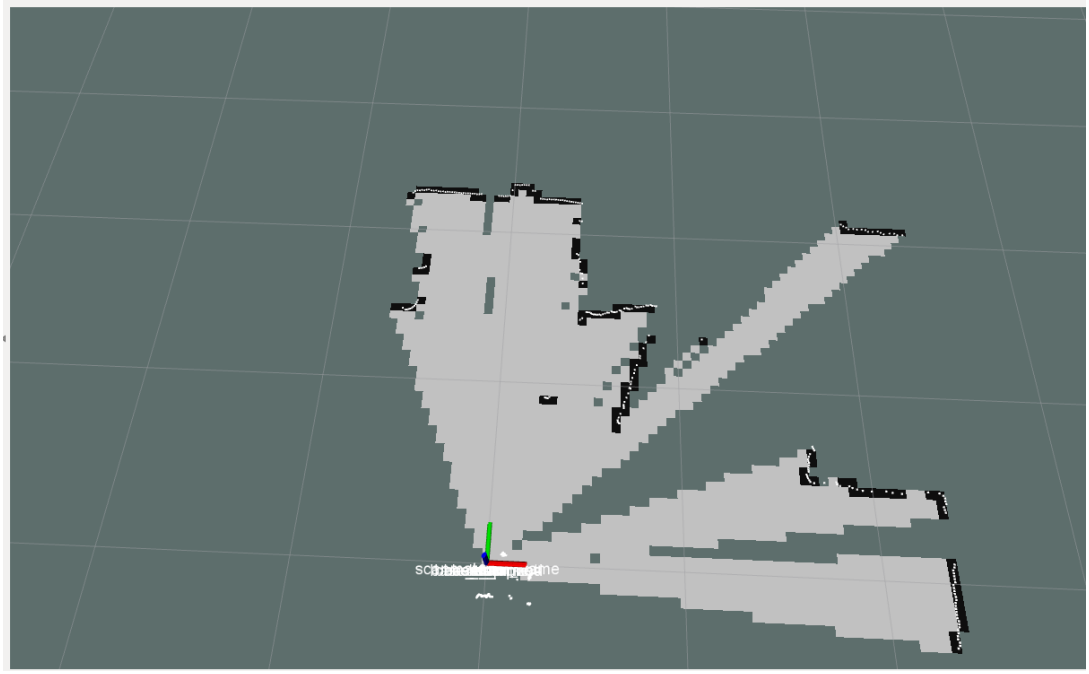

Figure 4-10 Visualized in Rviz, LIDAR data used by hector_slam to generate a map

The navigation commands were relayed to the wheelchair (which was still on the raised platform) for observing the system response which also turned out to be as expected. The LIDAR was also incorporated in this test and map generated by hector_slam using the LIDAR's data was as expected (a sample map is depicted in figure 4-10).

Chapter 5

CONCLUSION

A novel concept of short-range navigation and force-feedback has been developed and implemented in this thesis in the context of a semi-autonomous wheelchair. The system has been extensively tested using hardware in loop simulations. This chapter provides the conclusion and final thoughts for this endeavor, alongside with possible future work.

5.1 Final Thoughts

The ultimate catalyst for this work has been the desire to help and make the life of the people with disabilities who are wheelchair bound, better. This could help them in many ways such as reducing the number of accidental collisions, helping in navigating faster through obstacles etc. The unique force-feedback method of interfacing with the user makes us believe that it is the most appropriate and efficient means to provide information to the user without distracting them compared to other feedback methods.

Earlier works in this field either concentrated on force-feedback or on autonomous navigation. To the best of our knowledge, we are the first to bring both of them together with a balance between autonomy and manual control.

5.2 Future Work

Extensive user testing could be performed on the system to get appropriate feedback and make the respective improvements to the system. The force-

feedback effects can be optimized based on user experience feedback. This could be extended for use by people with visual impairments who are wheelchair bound to help them navigate independently.

References

[1] Kaye, H. S., Kang,T. ad LaPlante, M.P. (2000). Mobility Device Use in the United States. Disability Statistics Report. Washington, D.C.: U.S. Department of Education, National Institute on Disability and Rehabilitation Research.

[2] A. Hosseini, F. Richthammer and M. Lienkamp, "Predictive Haptic Feedback for Safe Lateral Control of Teleoperated Road Vehicles in Urban Areas," 2016 IEEE 83rd Vehicular Technology Conference (VTC Spring), Nanjing, 2016, pp. 1-7.

[3] Pillarisetti, A., Pekarev, M., Brooks, A. D., Desai, J. P. (2006). Evaluating the Role of Force Feedback for Biomanipulation Tasks. Proceedings of the IEEE Conference on Virtual Reality. Washington, DC.

[4] Wagner, C. R., Perrin, D. P., Howe, R. D., Vasilyev, N., Del Nido, P. J. (2006). Force Feedback in a Three-Dimensional Ultrasound-Guided Surgical Task. Proceedings of the IEEE Conference on Virtual Reality. Washington, DC.

[5] Feintuch, U., Rand, D., Kizony, R., Weiss, P. L. (2004). Promoting Research and Clinical Use of Haptic Feedback in Virtual Environments. Proceedings of Fifth International Conference on Disability, Virtual Reality and Associated Technologies (ICDVRAT'04). 141-147.

[6] Fattouh, A., Sahnoun, M., Bourhis, G. (2004). Force feedback joystick control of a powered wheelchair: preliminary study. Systems, Man and Cybernetics, 2004 IEEE International Conference on, 3. 2640-2645.

[7] Luo, R. C., Hu, C. Y., Chen, T. M., Lin, M. H. (1999). Force reflective feedback control for intelligent wheelchairs. Intelligent Robots and Systems, 1999. IROS'99 Proceedings. 1999 IEEE/RSJ International Conference on, 2. 918-923.

[8] Protho, J. L., LoPresti, E. F., Brienza, D. M. (2000). An Evaluation of An Obstacle Avoidance Force Feedback Joystick. Research Slide Lecture for Wheelchair University. University of Pittsburgh, PA.

[9] https://en.wikipedia.org/wiki/Wheelchair

[10] Richard C. Simpson, PhD, ATP (2005). Smart wheelchairs: A literature review. Journal of Rehabilitation Research & Development, Volume 42 Number 4. 423-438

[11] D. Innala Ahlmark, H. Fredriksson and K. Hyyppä, "Obstacle avoidance using haptics and a laser rangefinder," 2013 IEEE Workshop on Advanced Robotics and its Social Impacts, Tokyo, 2013, pp. 76-81.

[12] Matsumotot, Y., Ino, T., Ogsawara, T. (2001). Development of intelligent wheelchair system with face and gaze based interface. Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on. 262-267.

[13] Kuno, Y., Shimada, N., Shirai, Y. (2003). Look where you're going [robotic wheelchair]. IEEE Robotics & Automation Magazine, 10(1). 26-34.

[14] Rahib H. Abiyev, Nurullah Akkaya, Ersin Aytac, Irfan Günsel, and Ahmet Çağman, "Brain-Computer Interface for Control of Wheelchair Using Fuzzy Neural Networks," BioMed Research International, vol. 2016, Article ID 9359868, 9 pages, 2016.

[15] Walters, C. (1997). Cop a Feel….with Haptic Peripherals. Game Developer. November 1997.

[16] About ROS, http://www.ros.org/about-ros/

[17] ROS Core Components, http://www.ros.org/core-components/

[18] Manhattan distance, https://en.wiktionary.org/wiki/Manhattan_distance

[19] Move base, http://wiki.ros.org/move_base?distro=indigo

[20] Hardware-in-the-loop simulation, https://en.wikipedia.org/wiki/Hardware-in-the-loop_simulation

Biographical Information

Arjun Mani Gupta received his Bachelor of Engineering degree from Anna University in Chennai, Tamil Nadu, India in 2012. He will be receiving his Master of Science in Computer Engineering from The University of Texas at Arlington in Arlington, Texas in Spring 2018. His Interests include Robotics, Embedded systems and End to End Product Development.