Design of a Shared Control Interface for a Powered Wheelchair

by

Asset Malik

Submitted to the Department of Robotics and Mechatronics
in partial fulfillment of the requirements for the degree of

Master of Science in Robotics

at the

NAZARBAYEV UNIVERSITY

May 2023

Author .....
Department of Robotics and Mechatronics
March 15, 2023

Certified by .....
Almas Shintemirov
Associate Professor
Thesis Supervisor

Accepted by .....
Vassilios D. Tourassis
Dean, School of Engineering and Digital Sciences

Design of a Shared Control Interface for a Powered Wheelchair

by

Asset Malik

Submitted to the Department of Robotics and Mechatronics
on March 15, 2023, in partial fulfillment of the
requirements for the degree of
Master of Science in Robotics

Abstract

Powered wheelchairs significantly improve mobility of individuals with physical limitations, but their high cost and challenges associated with traditional joystick controls often limit their accessibility and ease of use. This thesis aims to explore alternative modern input methods for controlling a standard powered wheelchair aiming to redesing the system for shared control implementation. A ground-up approach is taken, covering wheelchair electronics modification, localization, path planning, and comparison of different input methods.

The research primarily focuses on the development and evaluation of a experimental prototype based on an Otronica Pulse 310 electric wheelchair with a ZED2 camera and ROS installed laptop. Localization performance was optimized using Simultaneous Localization and Mapping (SLAM) by combining the ZED2 camera with encoders. Caster wheel-aware path planning was achieved using Model Predictive Control (MPC) and potentiometers mounted directly on the axis. Various user input methods, such as touchscreen control and shared control, were investigated, which proved to be more comfortable and user-friendly than traditional joystick controls.

Despite the limited time available for implementation, the study provides a solid foundation for future research on wheelchair input methods, path-planning, and control systems. It also offers valuable insights for enhancing the lives of powered wheelchair users through improved control interfaces, shared control, and navigation technologies.

Thesis Supervisor: Almas Shintemirov
Title: Associate Professor

Acknowledgments

This thesis could not be possible without moral support from my friends and family. Thank you for trusting in my abilities and reassuring that master degree is worth it.

Contents

1 Introduction **13**

2 literature review **15**

2.1 Shared control 15

2.2 Model Predictive Control 15

2.3 Brain-Machine interface 17

3 Methodology **19**

3.1 Hardware setup 19

3.1.1 Wheelchair modifications 19

3.1.2 Mounting a depth camera 26

3.1.3 Encoders 27

3.1.4 Arduino 28

3.2 Software setup 29

3.2.1 ROS 29

3.2.2 Reading from Arduino 31

3.2.3 ZED2 SDK and RTabMap 32

3.2.4 Extracting wheelchair dymanics information 35

3.2.5 User interface 36

3.2.6 Version 3 38

3.2.7 Version 4 38

3.2.8 Version 5 - Unity and Quest 2 39

4 Results **41**

4.1 Localization 41

4.2 Caster wheel aware path planning 41

4.3 UX/UI 42

4.4 User review 42

4.5 Discussion and future work 43

5 Conclusion **45**

List of Figures

3-1 Setup diagram made for conference poster 20
3-2 Data packets captured on the data bus between JSM and VR2 22
3-3 Circuit for single wire UART communication 23
3-4 First prototype of Arduino wiring into JSM 24
3-5 Low pass filter for the Arduino PWM 25
3-6 Potentiometer on the caster wheel as reference. 26
3-7 ZED2 Camera located on the stand over the head looking down-forward 27
3-8 Encoder, mounted on the frame, on axis with motor 28
3-9 Simplified Dataflow of the wheelchair control. 30
3-10 Wheelchair's gazebo simulation with working joints. 30
3-11 Wheelchair's simulation with set up goal for the controller to drive to 31
3-12 Map of the 4th floor of C4 building with non parallel walls 33
3-13 ALARIS lab room, with one revolution path of robot around room . . 34
3-14 ZED2 internal IMU vs Jackal as reference. 34
3-15 Map inpection tool, to evaluate the map 35
3-16 Laptop with first version of motor control software 36
3-17 External display with a virtual joystick as input 37
3-18 ROS Navgoal as input method 38
3-19 Overlay of goal positions on the image 39
3-20 VR gaze as input 40

4-1 Photo of wheelchair in RDC right before testing with human patient. 43

List of Tables

3.1 Resolution and Framerate (fps) 32

4.1 Comparison of Sensor Technologies 41

Chapter 1

Introduction

The wheelchair is common way to increase mobility of people. Wheelchairs powered with motors are less common due to their speciality and can be a costly investment. In extreme cases powered wheelchairs are not even worth it, due to limited benefits and additional requirement from the user. Joystick control may require user finesse and create new complications. This project focuses on the motion of the wheelchair and ability of user to maneuver the vehicle.

Most of existing literature is focused in specific aspect of shared control for the powered wheelchairs. However, due to lack of a foundation, this thesis has a ground up approach, from wheelchair electronics modification, localization, path planning to comparison of different input methods.

The proposed research question was to identify which modern input methods are better suited to control the powered wheelchair, especially when the user lacks finesse and mobility to use traditional joystick. Throughout the research, more question have been found, including but not limited to control of proprietary VR2 hardware over rnet bus for research purposes and affect of the caster wheels on path planning and torque.

By answering mentioned questions, it will be possible to have more narrow focus in future research. The localization, control and reliability of the platform are essential and need to be researched by creating a working prototype, which is this case was based on an Otronica Pulse 310 wheelchair with ZED2 camera and a ROS installed

laptop.

Due to limited time given for implementation, this research does not fully answer to the research questions, but does provide enough information for others to learn and validate. The thesis consists from related works, methodology that includes both software and hardware solutions, results and a conclusion.

Chapter 2

literature review

2.1 Shared control

It is possible to blend the control data and the used input to control a wheelchair while also providing a tactile feedback the user over the joystick [9].

Because the human intention is not clear for the shared control algorithm, Partially Observable Markov Decision Process can be used to predict user intention and have a motion plan evaluated beforehand [4].

The safety of the driver can be increased by use of Model Predictive Control (MPC) to have a Shared Control between the driver and a vehicle with safety constrains in the MPC [12]:

$$\begin{aligned} y_r(x) + \frac{w}{2} \leq y_F \leq y_l(x) - \frac{w}{2}, \\ y_r(x) + \frac{w}{2} \leq y_R \leq y_l(x) - \frac{w}{2}. \end{aligned} \tag{9}$$

2.2 Model Predictive Control

Differential Drive robots are based on two powered wheels that control both direction and throttle on the vehicle, allowing to freely rotate in plate and move along circular path. Due to only two contact points with the ground, robots with differential drive

require third contact point or counterweight to balance, so most of the wheelchairs have pair of the castor wheels. This complicates the dynamics of the wheelchair and requires caster wheel aware control. To make robot follow the path MPC goal is set to follow the planned path, which can be accomplished by transverse feedback linearization. [8]

$$\frac{\mathrm{d}}{\mathrm{d}t}{}^{\mathrm{w}}\mathrm{W}\overrightarrow{\mathrm{W}} = v \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}, \frac{\mathrm{d}\theta}{\mathrm{d}t} = \omega \tag{1}$$

The data form the input, depending on the needs of a patient, the input of the wheelchair control can be filtered using a notch filter or similar [1].

$$H_{\mathrm{notch}}(s) = \frac{s^2 + \omega_0^2}{s^2 + Bs + \omega_0^2} \tag{2.1}$$

The MPC control techniques could be used on Neurological Disorders patience by applying sensor data from the electrooculographs and electromyographs. Because MPC can be used in both simple and complex systems with long delay and non-minimal phase this approach is quite popular, especially because MPC can also provide control for multiple input and multiple output systems. Resulting in the following general formula [11]:

$$\mathrm{x(k+1)} = \mathrm{A_d} \cdot \mathrm{x(k)} + \mathrm{B_d} \cdot \omega_{\mathrm{d}}(\mathrm{k}) \tag{1}$$

$$\mathrm{y(k)} = \mathrm{C_d} \cdot \mathrm{x(k)} + \mathrm{D_d} \cdot \omega_{\mathrm{d}}(\mathrm{k}) \tag{2}$$

Where x, y and ω are vectors that include wheel velocities, angular acceleration and desired velocities as inputs.

To increase the accuracy of LIDAR navigation in narrow indoor environment it is possible to implement Artificial Potential Fields (APF) into the MPC in addition with Normal Distribution Transform-Simultaneous Localization And Mapping (NDT-SLAM) to increase the awaireness of the surroundings, especially dynamic obstacles, such as people for point to point path planning. [6]

2.3 Brain-Machine interface

There is a method to identify human intention by monitoring brain activity using a Bayesian to have a shared control of the wheelchair. The method involves two layer decision model that chooses the direction when correct combination of commands is issued [2].

There is also a way to control the wheelchair using widely known visual technique with letters on the screen, but it is also possible to provide steady-state somatosensory evoked potential (SSSEP) to increase the potency and accuracy of classification in BMI wheelchair application [3].

The same p300 was already tested on wheelchairs in 2012 both as a joystick emulator and as a waypoint selector [5]. Together with an eye blinking the accuracy of the control drastically increases [10].

Chapter 3

Methodology

3.1 Hardware setup

The setup is based around the Ortonica pulse 310 commercial powered wheelchair, that has a built-in VR2 controller, two DC motors that are normally hold by eletronics breaks untill released eletronically, and joystick module (JSM) for user input. The wheelchair by itself is capable of transporting people around with good enough speed and controllability, but even an experienced driver might need some time to adapt to the joystick input, especially when riding in reverse. The batteries are two of Blue Dynamic 540, each holding a voltage of 12V and a capacity of 40Ah, which should be enough to roam around for a while.

3.1.1 Wheelchair modifications

Mounted laptop

The project began by making a laptop stand for the wheelchair. The idea was sketched on the notebook and then designed in Fusion 360 CAD. The CAD files (both 3D and drawings) were sent to the Technopark to be manufactured out of the aluminum metal sheet with laser cutter with additional bending of small parts but no welding. The whole construction is held by bolts and nuts on the wheelchair frame using wheelchair's adjustment bolt sockets. The laptop in question is acer nitro 15 with

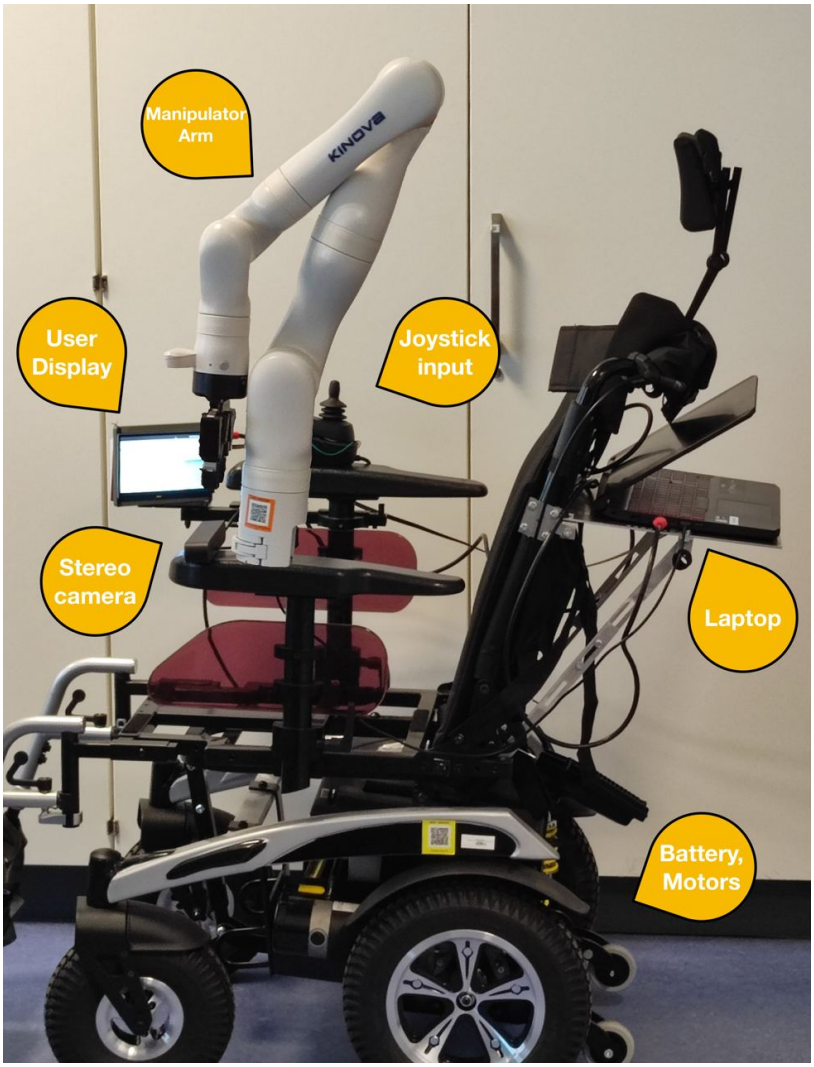

Figure 3-1: Setup diagram made for conference poster

intel CPU and nvidia graphics with CUDA capable compute acceleration. Laptop has a 65Wh battery and with average load of 20W it can plan and navigate the vehicle for more that an hour, which is enough for the proof of concept and can be extended by supplying charge to the laptop from the wheelchair's motor batteries, which would last for theoretical 48h. For the more optimized battery consumption the CPU's turboboost and hyperthreading features were disabled, because they only provided about x1.2 boost of performance for almost double the power consumption, which is redundant for the mobile robot that has less than half of CPU utilization. Same could be done with the GPU. By utilizing green with envy software, it was possible to limit the Thermal Design Power (TDP) limit of the GPU so it would be forced to use lower core clock speeds which are in the better efficiency range of the GPU.

Accessing the controller

The built-in VR2 controller is a good peice of equipment that does it job pretty good in a hands of typicall end user of the Ortonica wheelchair. As it has some specialized limitations premade. Automatic wheel lock, smooth speed increase and decrease are nice things to have. However, there were no dedicated external input for the PC to be connected to. It was not easy to just send the required speed to the VR2 and expect it to move accordingly. Reverse engineering the ports of the VR2 and JSM shown that it has external connection that could be used to control it, so we had to probe it using the oscilloscope and logic analyzer software.

The single cable had both the control signal and the feedback from the controller, which was captured and studied. The protocol used in the connection was not documented and not expected to be used by 3rd parties like us. There were online forums that tried to reverse engineer the signal, but those did not have the same setup and reasoning to work on it. Most of the Ortonica products used the CAN bus, studied by the community and that has some open-source examples of control. But our case was different, because our VR2 controller lacked RNET labeled port that would have the CAN bus, so it had to be reverse engineered.

Figure 3-2: Data packets captured on the data bus between JSM and VR2

The data captured on the JSM bus was initially though to be the CAN bus due to single cable implementation, however, the data did not have the structure of the CAN protocol and looked more like a universal asynchronous receiver / transmitte (UART) data. By tuning the parameters, the best baud-rate for this data was chosen to be 38400 bits per second and the bytes had no parity bits but included the checksum and the end of a packet.

Sending different control signals showed that the first HEX was a preamble. Second HEX dependent on the action from the JSM module and included encoded values for Speed Up, Speed Down and Horn buttons. Third HEX did not change the entire time, while forth and fifth were used as an axis of the joystick on the JSM. The checksum, which is the sixth and the last HEX in the packet was changing accoardingly.

Each full 6 hex packet send from JSM was replied by the VR2 controller over the same cable, with variable short random delay. The answer contaned another 8 hex of data in a packet which were not documented and could not be easily googled, leaving decoding as only option, that would take unknown amount of time.

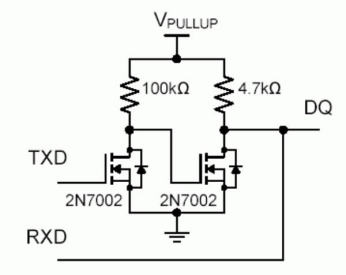

Figure 3-3: Circuit for single wire UART communication

The most difficult part when designing own JSM that would command over the VR2 was somehow using single wire as both receiver and transmitter. Especially when using very limiter oscilloscope with does not come with built-in logic analyser and has too narrow bandwidth when recording frames. The circuit built to communicate with VR2 (figure 3-3) failed to initiate the dialog in several different configurations and also provided no visible progress and time frame to finish.

The solution to the unknown protocol and wiring was to bypass it whatsoever and take advantage of existing circuitry by connecting the micro-controller directly to the

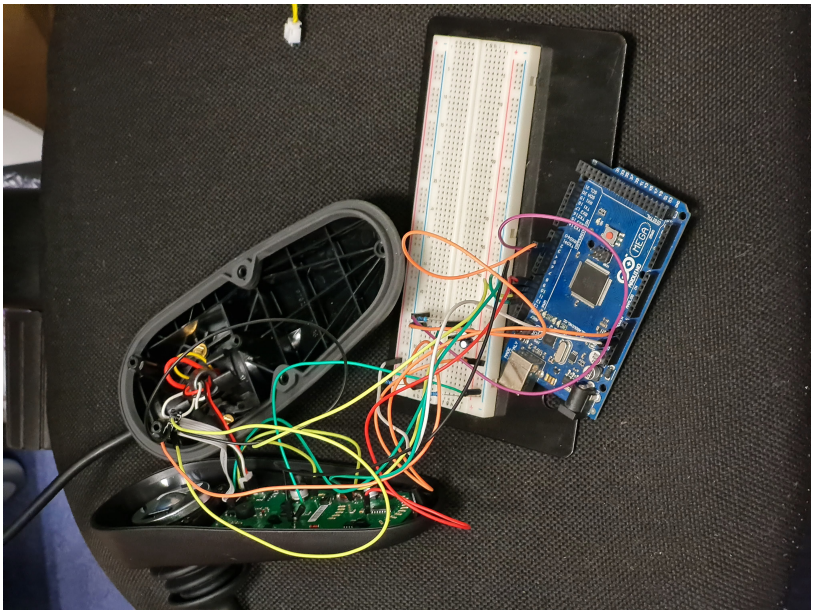

Figure 3-4: First prototype of Arduino wiring into JSM

button and joystick PCB traces. This would render JSM's joystick unusable but still could be used alongside original buttons. The Joystick had some proprietary 8 pin connector that required reference voltage and several ground points but was easily simulated with a Pulse Width Modulation (PWM) signal over low pass filter. But due to internal resistance of the joystick reading circuit, the filter output voltage was much lower than anticipated, which lead to linear distortion on the control signal, which can be easily compensated by linear mapping.

The AtMega328 micro-controller has good configurable PWM output that is clocked by the internal timer. Which can be set up to 62.5 kHz. After picking the suitable frequency, the filter was set up to have as low as possible peak to peak voltage with low enough bias offset. The capacitor and resistor of choise fell to values of $R = 20k\Omega$ and $C = 1\mu F$.

Caster wheel positioning

The caster wheel proved themselves to be important enough to consider then as an observable state of the system, because it directly affects torque threshold at the beginning of the motion depending on the angle of the caster wheel. That is why it

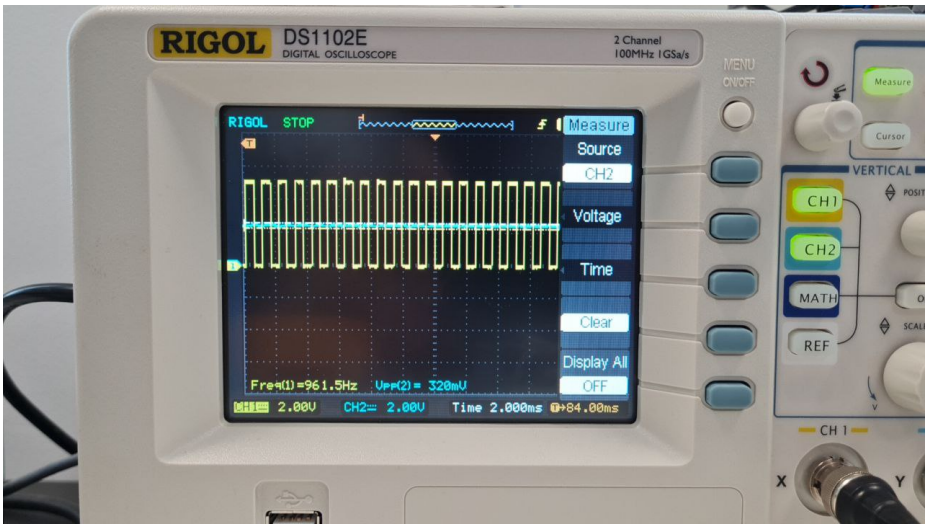

Figure 3-5: Low pass filter for the Arduino PWM

was important to have a real caster position reference both for MPC model and as a backup if model fails. To have a consistent sensing of castor wheel orientation it was decided to use the potentiometers directly on the axis. This approach is simple in design and requires small amount of additional hardware and software. Important aspect of this solution to consider was existance of a deadthone in the potentiometer, which can be solved by using another potentiometer with phase shifted orientation to cover the dead-zones of each other. The blending of the potentiometes values can be simplified by branching between the trusted sections of the linearized mapping of both sensors. The mapping algoritm is pretty simple and is a lookup table from pre-recorded reference values of raw sensor data and hard measurements of protractor.

$$f(x) = \begin{cases} g(x) & \text{if } x \in [a, b] \\ h(x) & \text{otherwise} \end{cases}$$

Where $g(x)$ is

$$g(x) = \begin{cases} y_i + \frac{x - x_i}{x_{i+1} - x_i}(y_{i+1} - y_i) & \text{if } x \in [x_i, x_{i+1}] \\ y_n & \text{if } x \geq x_n \end{cases}$$

and $h(x)$ is same but shifted by 180°.

The accuracy of such measurements might result in deviation of several degrees, but considered to be good enough because of the gradual enough increase of the torque (that allows in special cases to ignore the caster wheel altogether).

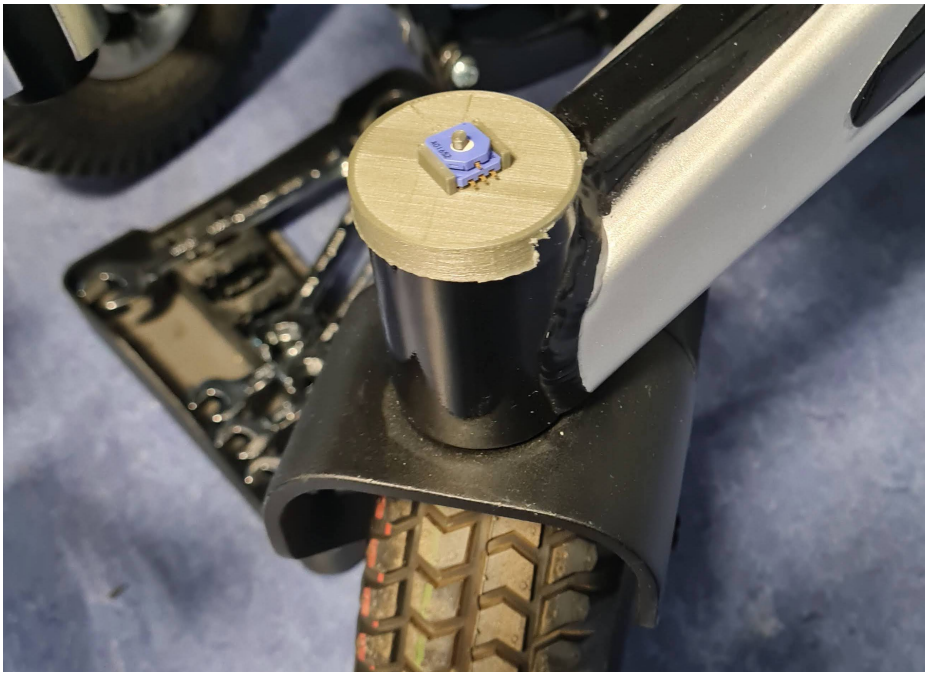

Figure 3-6: Potentiometer on the caster wheel as reference.

$$F_t = \frac{T_c}{r \tan \theta_c}$$

3.1.2 Mounting a depth camera

Despite ownership of several different depth cameral, out choise layed on the ZED2 stereocamera due to its higly documented software and SDK that allowed use of ROS with combination of RTabMap package to have a pointcloud map and the SLAM. To have a better view for the camera, it was mounted up above the wheelchair to have a forward top-down orientation and see the floor clearly. The camera itself has three mounting holes for the screws that were mounted on the custom 3D printed rotational joints. Whole construction was mounted on the aluminum extrusion for the height. The camera's location relative to the wheelchair's origin is measured and fed into the model of the wheelchair to be simulated in the virtual environment and better

localization. The mount itself is fixed good using metal parts and metal bolts on the metal frame, but the vertical aluminum extrusion provides noticeable oscillations when subjected to motion and due to external disturbances as wind. It might be possible to include such oscillations into the model, considering that the ZED2 also includes inertial measurement unit (IMU), but thought to be not major enough to cause noticeable error.

Figure 3-7: ZED2 Camera located on the stand over the head looking down-forward

3.1.3 Encoders

The ZED2 solution was proven to be noisy enough to consider additional localization techniques, so it was decided to use additional encoders mounted on the main wheels of the wheelchair. This on itself could be used for localization in deterministic enviroment with ready map and initial conditions, but it still needs realtime data from the camera to avoid collisions. The encodes were mounted in a way to be allined axially with the wheels, so 3D printed mounting parts were used to fix the encoders in place. The encoders of choise are AMT102. Those have great precision and accuracy. When

mounted they do produce some noise, but due to low signal to noise ration it looks pretty good. AMT102 have sync pulse and configurable tick count per revolution that was set to 1024 tick per rotation.

Figure 3-8: Encoder, mounted on the frame, on axis with motor

3.1.4 Arduino

The Arduino is a common interfacing solution that has enough ports to handle 4 potentiometers and 2 encoders that were mentioned in previous sections. Even the AtMega328 chip located in the lower spec options of Arduino can do the job, so we used one. The analog to digital converter (ADC) resolution of 10 bits provide good enough accuracy of 0.35° and the included 2 hardware interrupt pins are good enough for the encoders. The data is passed to the PC over the Universal Asynchronous Receiver/Transmitter (UART). Because of low reliability and data rate of the standard UART protocol, additional preamble and checksum were added into the data-frame of the packets. The data itself is described by a single C struct and is define on both

sender and receiver code.

```
struct SEND_DATA_STRUCTURE{
  unsigned long time;
  long tiks1;
  bool dir1;
  long tiks2;
  bool dir2;
  int  cast1;
  int  cast2;
};
```

By using structs instead of traditional Serial.print library it is possible to have less data sent, so the frequency is high enough for reliable localization (over 200HZ in preliminary testing). The data packet includes both caster wheel positions and encoder values with additional logical bits that include the direction of motion of the wheels. Which is redundant, but was useful at some point in time.

3.2 Software setup

The main framework of the project was decided to be Robotics Operating System due to its large codebase with community packages and great navigation stack. Previously mentioned ZED2 camera has a ROS wrapper with example launch-file to use and existence of RTabMap makes making Simultaneous mapping and localization easier. As for the Arduino part of the hardware, it was programmed in its self-titled development environment using wiring C. The user interface is done on the Golang due to my familiarity with the language, great concurrency model and quality of life features it provides. All software is handled by the laptop mounted on the wheelchair.

3.2.1 ROS

Robotics Operating System's (ROS) navigation stack is well suited for application as wheelchair. It heavily relies on the packages from the debian's apt package manager,

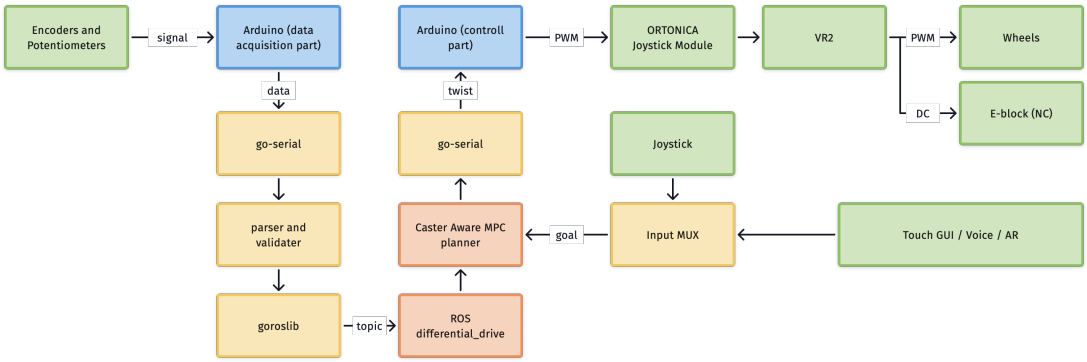

Figure 3-9: Simplified Dataflow of the wheelchair control.

so each ROS version is strictly tied to its Ubuntu distro release.

Gazebo simulation

Using tape measurements simple 3D model of the driving part of the wheelchair was modeled. The main focus of the model was the driving wheels with caster mechanism.

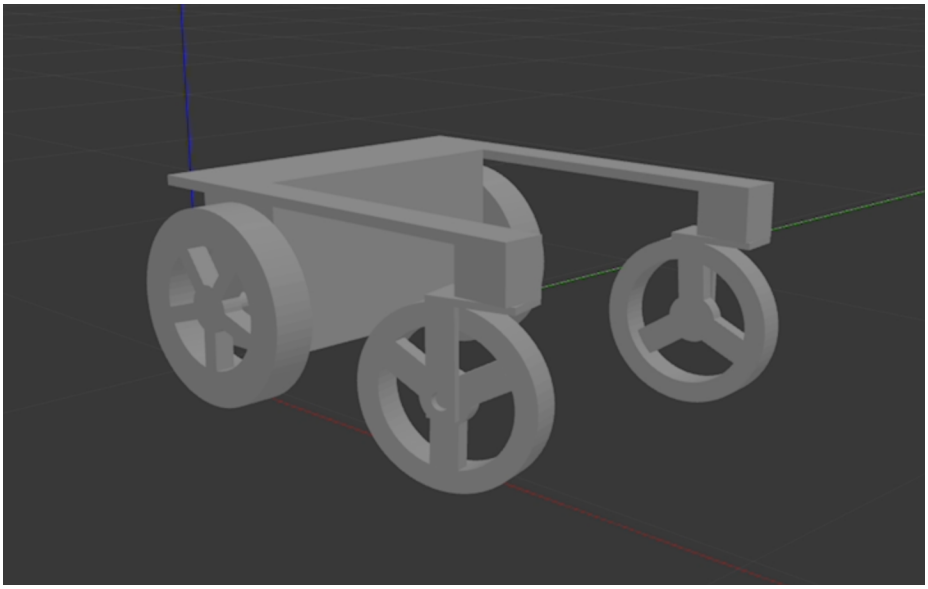

Figure 3-10: Wheelchair's gazebo simulation with working joints.

The URDF of the robot was generated using export tool for the Fusion 360 3D CAD software. The ready open-source script auto-generated the rotational joints on the corresponding joints in the fusion 360 assembly file. The model has joints on both

main wheels, the castor wheel and on the caster joints to simulate and capture caster position.

Differential Drive controller

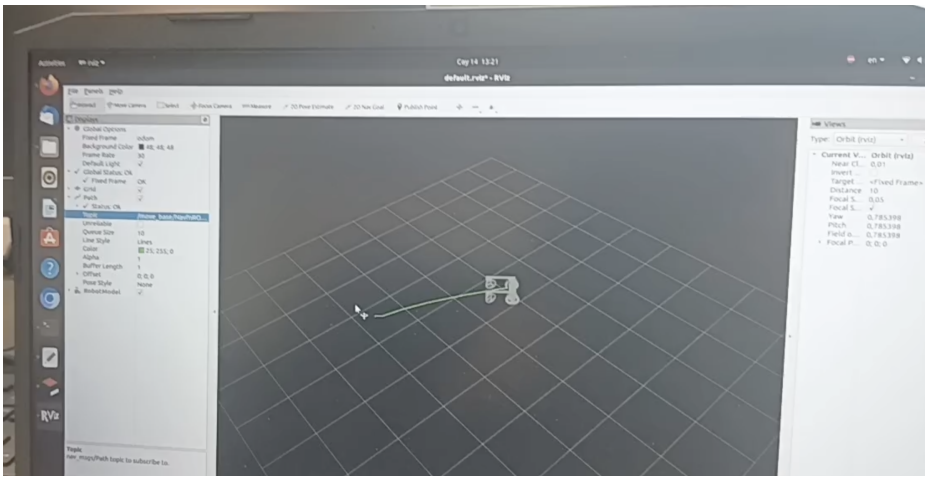

Figure 3-11: Wheelchair's simulation with set up goal for the controller to drive to

After setting up the acceleration and velocity parameters from the real life experiments, it was possible to create the controller for the wheelchair using ROS's navigation stack.

3.2.2 Reading from Arduino

The data packet that contains caster wheel and encoder data from the Arduino that arrived over UART is encoded using open-source EasyTranser library. Received data has to be published to the ROS navigation stack as a topic. It is possible to do so with a single binary written in Golang. Using go-serial data from UART is read, parses and checked before being published to the ROS using goroslib. This is an easy solution for the simple code that does not need to rely on other ROS packages, and it can easily be run by a roslaunch file.

3.2.3 ZED2 SDK and RTabMap

ZED2 camera is supplied with a great SDK that includes automatic 3D pointcloud reconstruction from stereo-camera video footage at different resolutions and framerates. It also provides a ROS wrappers for further use in the RTabMap package that can simultaneously localize robot and map the location. Most of the SLAM algoritms and solutions suffer from the "kidnapped Robot" problem, which makes robot hard to localize on previously made map if case of discontinuity in the position. This was especially relevant when the wheelchair was equipped with the lidar scanner instead of ZED2. Moreover, due to lidar's use of a single plane slice of the scene, it was hard for algorithms to distinguish different positions of the robot in the same corridor, due to it looking the same for the whole duration of corridor.

Resolution	Framerate (fps)
2208x1242	15
1920x1080	30
1280x720	60
672x376	100

Table 3.1: Resolution and Framerate (fps)

software environment

Even though the SDK has a great documentation, it is worth mentioning that the initial setup experience is far from ideal. The ZED2 camera SDK is built upon the infamous Nvidia CUDA parallel computing framework, which does lack native Linux support required by the ROS. This lead to several failed attempt to install all the required dependencies that would not conflict with each other. The SDKs have specific requirements for both Nvidia driver and CUDA version, that don't always work together perfectly. The automatic installer does not cover all the edge cases, leading to manual downloading and installation sequence with trial and error of different versions until everything works. There is an option to use precompiled docker version of SDK, but still it cannot manage to install the kernel modules for

the nvidia and CUDA drivers, so problem persists.

Limitations

When first trying out the solution the team has realized that sometimes when creating a map, the robot would make imperfect walls that do not go perpendicular and have some circularity to them, leading to gaps in the map topology that prevents it from being enclosed. Stitching process for the map creates distortions. Yet still, the error of the map is small enough to be used in this application and takes several tries to perfect it.

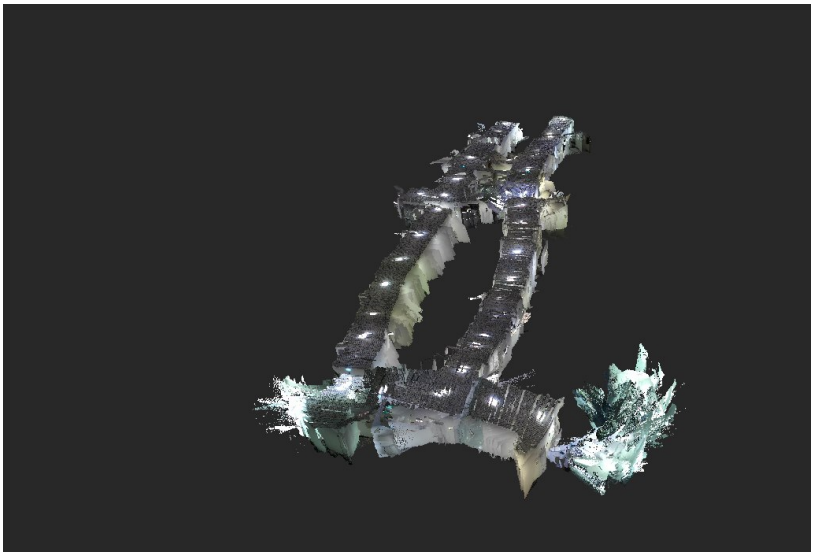

Figure 3-12: Map of the 4th floor of C4 building with non parallel walls

Test have shown that the RTabMap with ZED2 perform better when subjected to the normalized lighting with small dynamic range that fits into the tight window of shutter speed when setting up the camera parameters. The SLAM update rate, especially in low light conditions drops down drastically, and are almost unusable, providing noticable noise. It was decided to cover all natural sources of light and cover transparent doors with papers to increase the camera's ability to SLAM. Considering that the floor material is highly reflective and provided false feature points for the SLAM, the effect of complex lighting can seen on the figure 3-12. The performance of the camera was increased by making the camera angle positioned more toward the

ground from high above, providing top-down perspective that also includes several forward meters to avoid collisions. High resolution results is less noisy map but decreases the sample rate. Minimal resolution with maximum sample rate reduces low light performance, so the optimal setting are somewhere in between.

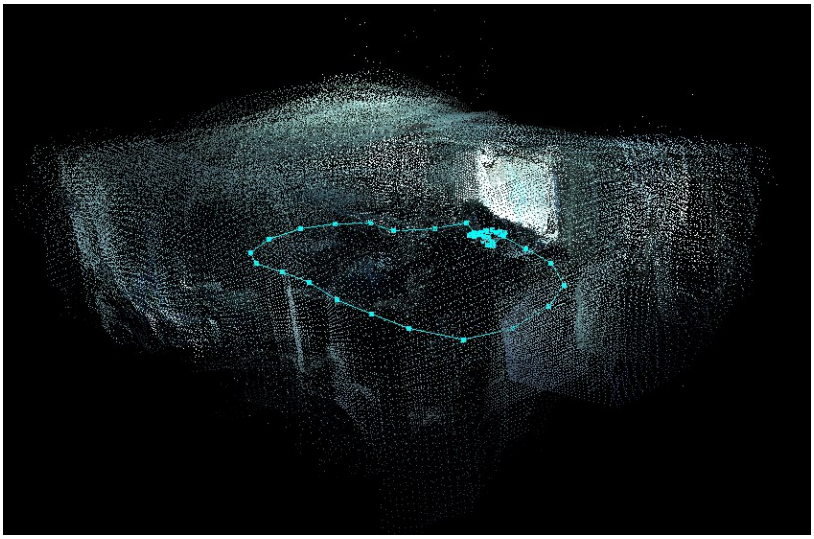

Figure 3-13: ALARIS lab room, with one revolution path of robot around room

It is possible to create the map of the location on the fly and then reuse said map to create waypoints. In case of the wheelchair, most used locations like bedroom, kitchen and toilet, can be saved to be used daily upon request.

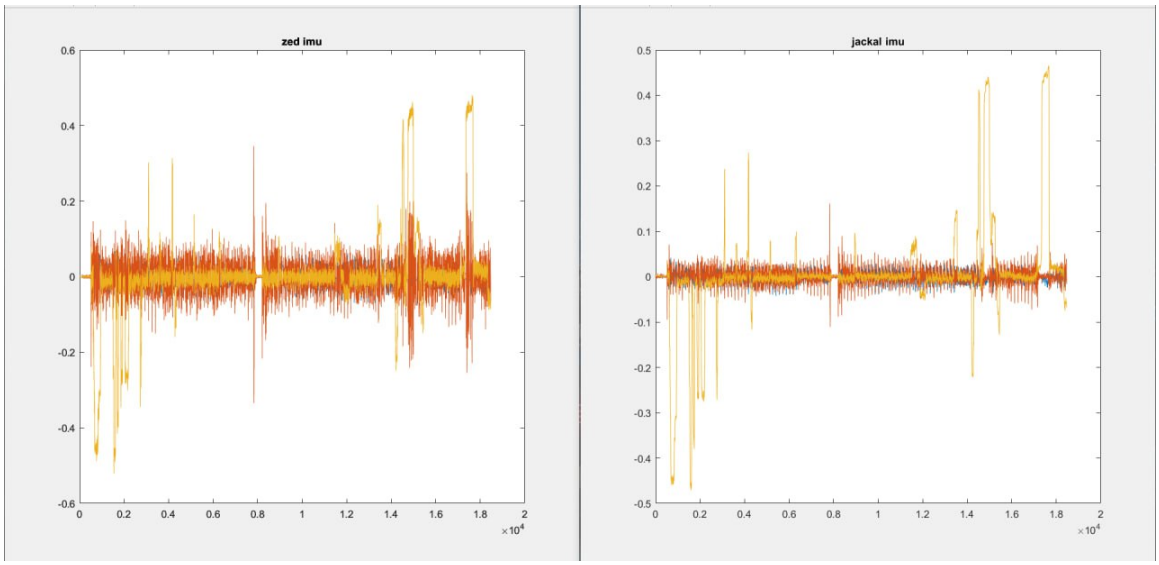

Figure 3-14: ZED2 internal IMU vs Jackal as reference.

The ZED2 data was deemed to be too slow and noisy to be used as main localization tool, leading to alternative solutions. Previously used lidar was not considered due to previous issues with localization in spaces like long empty corridors that have almost no identifying features for lidar. The ZED2 provides internal IMU sensor, but it too was too noisy, this forced us to have an encoder pair on each of the wheels that was mentioned in the Hardware part of the methodology.

3.2.4 Extracting wheelchair dymanics information

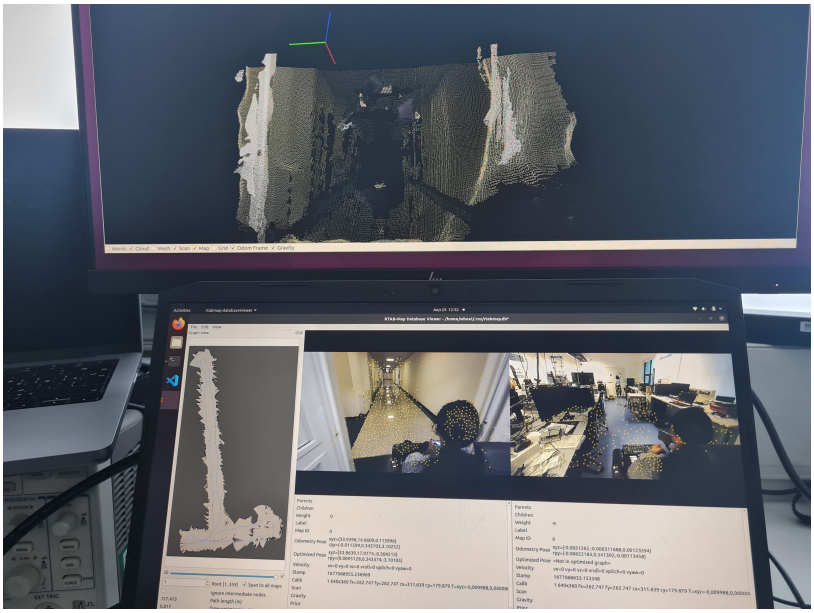

Figure 3-15: Map inpection tool, to evaluate the map

The ZED2 and RTabMap provide enough data to evaluate the instantaneous speed and acceleration of the robot when controlled in open-loop fashion for future system modeling. This information is especially important for the MPC. That is why the generated ROS odometry and positional data were stored in the rosbag file to then be analyzed by the software. It is possible to view rosbag files using both python and Matlab. Both of which were used in several occasions for different data. Even though the Matlab ROS toolkit is easier to setup, due to not known reasons, it performed far longer so that the python solution was preferred. The python code resulted in the maximum velocity of $1.25m/s$, which is a bit slower than what is stated on the

datasheet, but might be due to a mass of a user, which at that time was me with a weight of 69kg. The acceleration is around $1.042 m/s^2$, which is rather slow, but good enough.

3.2.5 User interface

Version 1

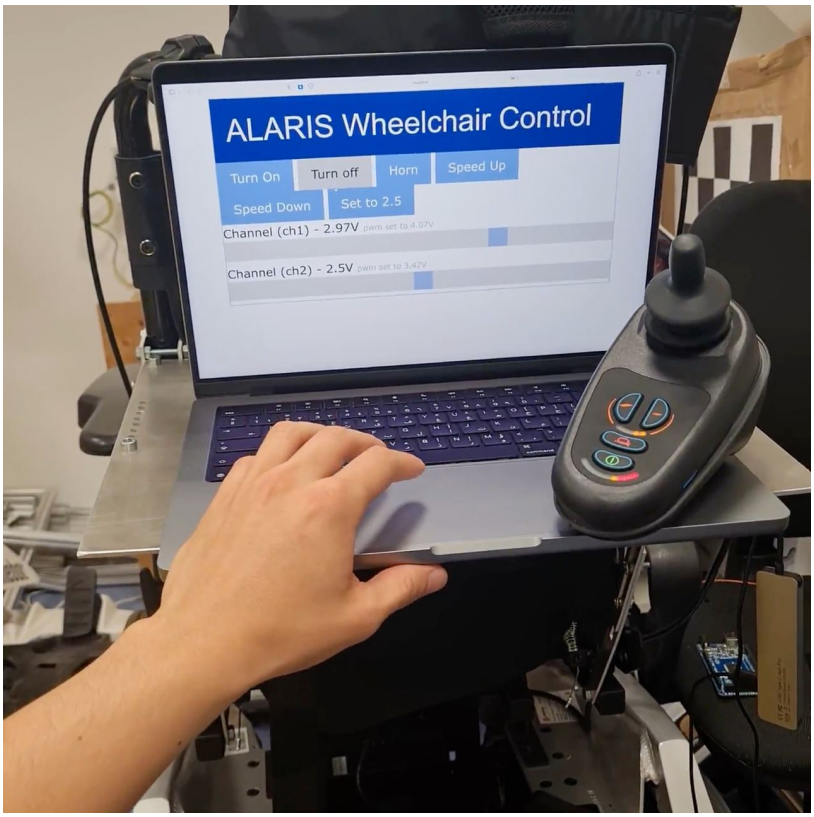

Figure 3-16: Laptop with first version of motor control software

Initially, when the wheelchair dynamic model was not ready, to test out the motors and to control them from the software, simple graphical user interface (GUI) was constructed. This first version of GUI replicated most basic functionality of the JSM with its speed, power and horn buttons. The GUI also had two sliders for joystick axis control. Due to inclusion of the low pass filter into the control circuit, the GUI also had a visible voltage data on the axis sliders. This GUI was used to test the direction and angular velocity of the motors and to make simple relocations

of the heavy wheelchair. This GUI was also helpful to get wheelchair's maximum acceleration data for the dynamic model. The software is written in Golang with go-serial to communicate with the wheelchair's JSM and VR2 controller circuits over Arduino. The control signal to the Arduino is commanded from the ROS topic. That can be published with the GUI's sliders. The data is not sent directly and loops form the ROS to be captured as a topic into the rosbag to the data acquisition and analysis.

Version 2

Figure 3-17: External display with a virtual joystick as input

To make controlling the wheelchair from software easier the setup was equipped with an external display that has a touchscreen. The GUI was updated to include a virtual joystick that publishes its state to the topic that is initially was send to the controller over UART bypassing the MPC. The same topic is also used to be blend with a shared control path planning to avoid collisions. This input method was ready when we had a trip to the Republic Diagnostic Center (RDC), which was tested by

the child, adult and medical Doctor and was generally though to be good enough, and was even considered easier to use by the child over the traditional Joysticks.

3.2.6 Version 3

The third version of the control that utilizes the ROS's goal position setter together with the visual stimulus letters from the typical Brain Machine Interface implementation. That would trigger the path planning and movement based on the selected letter on the map. The letters are positioned on the touchscreen of the wheelchair connected to the laptop. The screen shows previously made map with manually placed waypoints that have random single characters assigned to them to be used as a reference for the BCI, voice activation or simple touch. The final orientation of the wheelchar is also saved into the waypoint to ease the use. Manual and shared control are still accessible as an option when the waypoint is not set yet, or when needed.

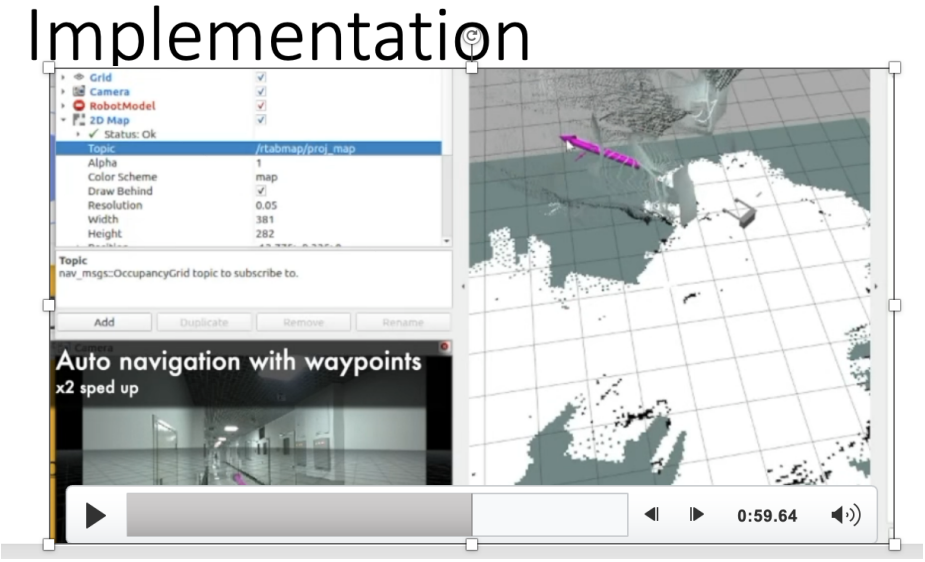

Figure 3-18: ROS Navgoal as input method

3.2.7 Version 4

Latest version that was drafted is a same map that has letters on it but it is an overlay over a camera footage that is powered by the landmarks. The landmarks are

positioned by the simple A4 papers with QR markers on them. Current draft uses AR Track ALVAR ROS package to track the position of the markers to then create a path to it, but to visualise the letters on the markers the AR version of software made in Unity uses the EasyAR asset prefab. Due to having direct gaze as an input, it is possible to use the AR with only the gaze without reliance on the BCI.

Figure 3-19: Overlay of goal positions on the image

The figure 3-19 shows early prototype of overlay made using 3D scene drawn in Blender with depth data from the room imaging and manually positioned landmarks over the tables.

The letters act as a reference for waypoints that can be used as a keywords for voice activation or BCI.

3.2.8 Version 5 - Unity and Quest 2

The actual implementation of the VR solution turned out to be easier to implement than initially intended. The Oculus Quest 2 headset provides great Software Development Kit (SDK) that contains access to headset's internal positioning system, that can keep track of so called "Spacial anchor points" that can be Unity game object that is registered to the database of the headset that can store up to 1500 items in the

room. This objects remember their relative position in space and can be easily saved and loaded when the headset identifies its position in the pre-defined environment. This requires the user to manually map the location to setup it once, but will persist for a while.

Figure 3-20: VR gaze as input

Chapter 4

Results

4.1 Localization

We went from using lidar to ZED2 stereo-camera and to additional encoders. Localization performance was the best when combining the ZED2 camera with encoders. This allowed to have Simultaneous Localization from encoders and Mapping from the ZED2 camera, that also provides dymanic obstacles like humans.

Table 4.1: Comparison of Sensor Technologies

Name	**Accuracy**	**Comments**
Lidar	±3cm	fails in corridors
ZED2	±0.4cm	fails with transparent objects and light sources, noisy
ZED2 & encoders	±0.4cm	good enough

4.2 Caster wheel aware path planning

Is was possible to track the caster wheel position using the potentiometers mounted directly on the access as the reference. There is currently no graph showing the tracking performance of the MPC observer to be show, but it will be in the next draft. The path planning for the wheelchair works good, can avoid obstacles and finishes in correct specified orientation as seen on figure **??**.

4.3 UX/UI

Each iteration on the Grahical User interfaces provided more positive feedback and indicated right development progression. The labmates who previously could not or were afraid to drive the wheelchair now had fun of driving around the lab and corridor. The untested AR version of the interface can be the next thing to validate and looks very promising. The amount of giver user inputs allows as BCI, gaze and voice allowes hands free action while more precise control can be initiated with shared control touchscreen or joystick input, while still providing manual control for special cases.

4.4 User review

Professor Anara Sandygulova has provided an opportunity to test out the wheelchair performance with an end user that has currently residing in the Republic Diagnostic Center (RDC) of Astana. After negotiating with patien's physician, parents and himself, the boy tried out the wheelchair. His initial reactor was cautios, but after getting used to the controls he started racing around the hall and was joyfull. Mother of boy shared her experience while the kid was riding and said that such autonomous vehicle would help her out by transporting her child around the facility and between the procedures without her full involvement. The kid has shared that he enjoyed the controls and that the touch screen interface is more easier to use than traditional joystick in powered wheelchairs. The main concern of the boy was that there is no way to see behind the chair and it would be great to add a camera or parktronic. It is also had for the boy to reach things that might fall to the floor, so the boy discussed the possibility of adding the Manipulator to the wheelchair to access things beyond reach.

Figure 4-1: Photo of wheelchair in RDC right before testing with human patient.

4.5 Discussion and future work

This thesis work is a part of Professor Anara Sandygulova's and Professor Shintemirov's grant that also includes the manipulator control part that would be great to implement and would cover the needs of a person mentioned in the User review part. The proposal for this project included conparison between different input methods including VR, AR, voice and BCI, but due to limit in time, the project did not cover this aspect of the problem. One of the sources has had a input analysis for wheelchair patience with Parkinson's Tremor, which was classified and filtered using clever techniques, which could be also incorporated into this study bit was not [7].

Chapter 5

Conclusion

The research questions stated in the introduction part of the thesis were:

- Which input methods are better suited to control the powered wheelchair?
- How to control the proprietary VR2 hardware?
- To what extend do caster wheels affect the navigation on wheelchair?

User test have shown that the touchscreen mounted on the wheelchair does provide more comfort than the traditional joystick when both driving manually and while shared control, although the learning curve is smaller, there is still some training required, especially for those who are already used to the joystick.

The VR2 controller proved itself to be complex enough to be bypassed and not researched fully. The work has begun. There is no way to know how much time would it take to reverse engineer and if it is even worth it, but it seems possible.

It is possible to use the wheelchair without inclusion of caster wheels into dynamic model and just reduce it to the differential drive model, but there are some special cases when caster wheel are close or more than perpendicular and when the motion is affected enough to cause unexpected torque requirements and provide huge path error of the robot, which can be minimized using Model Predictive Control to have a caster wheel observer or by directly mounting sensors on the caster wheels.

This thesis does provide enough groundwork for future research on input methods and wheelchair path-planning and control methods and may positively impact lives of powered wheelchair users in near future.

Bibliography

[1] Jon Arrizabalaga, Niels van Duijkeren, Markus Ryll, and Ralph Lange. A caster-wheel-aware MPC-based motion planner for mobile robotics. In *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE, December 2021.

[2] Xiaoyan Deng, Zhu Liang Yu, Canguang Lin, Zhenghui Gu, and Yuanqing Li. A bayesian shared control approach for wheelchair robot with brain machine interface. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 28(1):328–338, January 2020.

[3] Keun-Tae Kim, Heung-Il Suk, and Seong-Whan Lee. Commanding a brain-controlled wheelchair using steady-state somatosensory evoked potentials. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(3):654–665, March 2018.

[4] Ayse Kucukyilmaz and Yiannis Demiris. Learning shared control by demonstration for personalized wheelchair assistance. *IEEE Transactions on Haptics*, 11(3):431–442, July 2018.

[5] Jinyi Long, Yuanqing Li, Hongtao Wang, Tianyou Yu, and Jiahui Pan. Control of a simulated wheelchair based on a hybrid brain computer interface. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, August 2012.

[6] Hibiki Matsuura, Kenichiro Nonaka, and Kazuma Sekiguchi. Model predictive obstacle avoidance control for an electric wheelchair in indoor environments using artificial potential field method. In *2022 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, January 2022.

[7] Richard Meyer, Fabian Just, Raymond A. DeCarlo, Milos Zefran, and Meeko Oishi. Notch filter and MPC for powered wheelchair operation under parkinson's tremor. In *2014 American Control Conference*. IEEE, June 2014.

[8] Viet Thuan Nguyen, Chouki Sentouh, Philippe Pudlo, and Jean-Christophe Popieul. Path following controller for electric power wheelchair using model predictive control and transverse feedback linearization. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, October 2018.

[9] Viet Thuan Nguyen, Chouki Sentouh, Philippe Pudlo, and Jean-Christophe Popieul. Joystick haptic force feedback for powered wheelchair - a model-based shared control approach. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, October 2020.

[10] Dilok Puanhvuan and Yodchanan Wongsawat. Semi-automatic p300-based brain-controlled wheelchair. In *2012 ICME International Conference on Complex Medical Engineering (CME)*. IEEE, July 2012.

[11] Norberto Scarone, Gustavo Monte, Ruben Bufanio, Damian Marasco, Ariel Agnello, and Pablo Liscovsky. Electric wheelchair sensing and control algorithms for users with neurological disorders. In *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, October 2021.

[12] Linhuan Song, Hongyan Guo, Fei Wang, Jun Liu, and Hong Chen. Model predictive control oriented shared steering control for intelligent vehicles. In *2017 29th Chinese Control And Decision Conference (CCDC)*. IEEE, May 2017.