Development of Waterloo Robotic Rollator (WATRR)

by

Abdullah Rashid Yeaser

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2021

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

One of the major risk factors for impaired mobility is aging, and with the aging population on the rise, the demand for assistive technologies for individuals with mobility impairment is at an all time high. Impaired mobility can lead to loss of independence, increased chance of mortality, deterioration of health, decreased cognitive function and a poor quality of life. Moreover, individuals with impaired mobility also tend to have higher hospital utilization costs. Mobility capability can be (re)built through the use of assistive technologies.

Rollators/Walkers are a commonly used mobility aid that has shown to help with mobility by providing support, particularly transferring a portion of the lower limb loads to the upper limbs. However, safety has been a concern with rollators, with thousands of accidents occurring every year. Currently, many research projects are investigating methods to improve rollators, particularly surrounding the use of robotic rollators.

At University of Waterloo Neural and Rehabilitation lab (NRE lab), our goal is to develop technology to improve lives of people, with development of robotic rollators being one of our research foci. The Waterloo Robotic Rollators (WATRR) is an active rollator system with built-in sensing and actuation systems. It is believed that the user experience and safety of rollators can be improved through the use of smart control algorithms.

The purpose of this thesis was to develop methods to address safety and user experience concerns by proposing a hybrid control approach, where distance and orientation control are key control parameters, including automatic braking. First, the Waterloo Robotic Rollator (WATRR), a low weight robotic rollator platform, representative of current rollators with sensors and actuators is presented. I describe key design decisions for the platform, offer an overview of the software architecture, and discuss further research development goals. The proposed hybrid controller is then described and both simulation and experimental data for controller design is presented.

To enable the envisioned hybrid control systems, a human state estimator and a robot state estimator are required. The human state estimator uses computer vision and machine learning in a hourglass network structure to predict shoulder locations. Using the estimated location and depth data, human velocity, distance and orientation relative to the rollator are estimated. For the robot state estimator, a new velocity estimator based on learning methods is proposed. As rollator lateral velocity can be difficult to estimate with traditional methods, we propose an augmented learning-aided state estimator. This estimator is a Long- Short-Term Memory (LSTM) based estimator, augmented with an Unscented Kalman Filter (UKF). The proposed estimator was validated through experimental data.

The main contribution of this thesis was a new lightweight rollator system with sensors and actuators that enabled development of advanced controls. Next, previous control systems are not only improved upon by using a new hybrid controller but also implemented on our platform. A new robot state estimator is developed that relies solely on the kinematics and is able to estimate lateral velocity with a mean error of $< 10mm/s$ without requiring additional instrumentation or knowledge of the rollator's time varying parameters. Finally, a new human state estimator is designed which does not require instrumentation on the human and outperforms current estimators.

Acknowledgements

First and foremost, I would like to thank my amazing supervisors Dr. James Tung and Dr. Ehsan Hashemi for their continuous support and guidance. A huge shout out to my NRE lab mates for all their help.

I would also like to thank my family and friends for their love and support over the past two years. This would not be possible with them.

Dedication

This is dedicated to my parents.

Table of Contents

List of Figures

List of Tables

1 Introduction

Mobility is defined as "the ability to move or be moved freely and easily" and is considered an important aspect of an individuals life as it is directly related to participation and quality of life [3, 4]. One of the major risk factors for impaired mobility is aging, and with the aging population on the rise, the demand for assistive technologies for individuals with mobility impairment is at an all time high [5,6]. It is expected that by 2051, the population of Americans over the age of 65 is expected to be double than in 1999 [6]. A study released by CDC [7] estimated that 25.7% of the American population have some form of disability, with mobility disability being the most prevalent at 13.7%.

Impaired mobility has many adverse effects on an individual's quality of life. Impaired mobility leads to loss of independence, decreased cognitive function, deterioration of health, increased risk of mortality and increased risk of falls and fractures [4,5,8]. Individuals who have lost their mobility also tend to have higher healthcare utilization and associated costs [4]. In Canada, the average cost of healthcare for individuals under the age of 65 is $\$2,341$ annually and increases to $\$20,387$ for those over 80 years [9]. These effects can be partially mitigated through building mobility capacity and remaining physically active through the use of assistive technologies.

Mobility aids allow users with low to moderate impairment to remain active by helping reduce lower limb loading crucial for individuals with limited capability, lower limb weaknesses, or injuries [10]. Many devices, such as rollators, improve balance and stability among users [10]. Mobility aids also have psychological and social benefits such as improved confidence, feeling of safety, and being able to remain in the workforce [10]. Physiological benefits include prevention of osteoporosis, increased cardiovascular health, and improved venous circulation [10].

The three main forms of assistive devices used for mobility are wheelchairs, canes, and walkers. As with any device, each form possesses benefits and drawbacks. While wheelchairs provide mobility to those with little–to-no lower limb function, wheelchairs fail to promote lower limb exercise, thus impacting the health benefits associated with physical movement including reduced risk of cardiovascular disease, diabetics, stroke, colon and breast cancer [11]. Prolonged sitting on wheelchairs can cause muscle weaknesses, pressure sores, and mental stresses [12]. While wheelchairs have many negative effects, it is often the only choice for users with severe impairment. On the other hand, canes are suitable for individuals with low level of impairment and facilitate lower-limb activity. However, canes provide limited support and stability. Walkers (and rollators) are a compromise of both forms, with greater support and balance than canes, but still permit lower-limb use

for mobility. To address fatigue issues, walker/rollator users can sit down and rest when required.

Walkers are simple rigid structure with legs which provides support to the user. Rollators are a form of walker, with wheels in its legs which helps reduce the energy required for ambulation by avoiding upper limb demands associated with picking up the walker for each step [13]. Rollators also allow the user to walk in their (near) normal gait pattern which is a key benefit. Drawbacks of rollators are dominated by safety risks, including increased risk of slips and falls due to improper use, tripping, collisions while transversing through doors, and the rollator rolling away from the user [14, 15]. Stevens Et. al [15] reported that there are approximately 47,000 injuries annually related to accidents using mobility aids, 87.3% of which were associated with walkers. Riel Et. al [16] reported that another issue of rollators are that 30%–50% of older adults abandon their device soon after receiving them due to difficulty to use and safety risks, with weight being one of the main complaints. These reports indicate there is a need to improve the design of rollators to improve safety and user experience.

Robotic rollator solutions have been proposed to address a subset of the aforementioned issues. Some of the main areas of research for robotic rollators include rollator control stability and motion support, navigation and localization, monitoring and safety [12,17–20]. A fundamental issue with prior research projects is a lack of experimental validation data and unrealistic real world implementations due to bulk and/or expense (i.e. having the user wear sensors, using $\geq$ \$500 force sensors). The purpose of this thesis was to design a new low-cost user friendly research platform, develop controllers to improve safety and user experience, develop new robot and human state estimators to enable the proposed controllers and implementation of the proposed control systems.

The rest of this thesis is structured as below:

- Chapter 2 covers the design of our rollator and the instrumentation used for the various sensing requirements.
- Chapter 3 describes the overall control system approach for the rollator. The purpose of this chapter is to review current control systems, develop new approaches, and present control results.
- Chapter 4 details development of the proposed Human State Estimator. The purpose of this chapter is to develop a system that is able to estimate the current user state which is used in the control system.

- Chapter 5 covers the proposed Robot State Estimator. This chapter covers current approaches to robot state estimators and introduces a novel approach to estimate robot velocity using a learning based augmented method. It also overviews current estimators which are also used in this thesis.
- Chapter 6 summarizes the findings from this research and future work proposed.

2 Design of WATRR

To enable robotic control, sensors, instrumentation and software implementation are required. The NRE lab had a research platform, seen in Figure 1a which was bulky and heavy, similar to many other research platforms, making it difficult to use for real-life representative experiments. It also did not have all the features (i.e. no RGB-D camera, properly sized motors) required to enable the proposed controls. As the performance of the control system is heavily dependent on the sensors and the platform itself, the current platform as seen in Figure 1b was simplified to better reflect actual use.

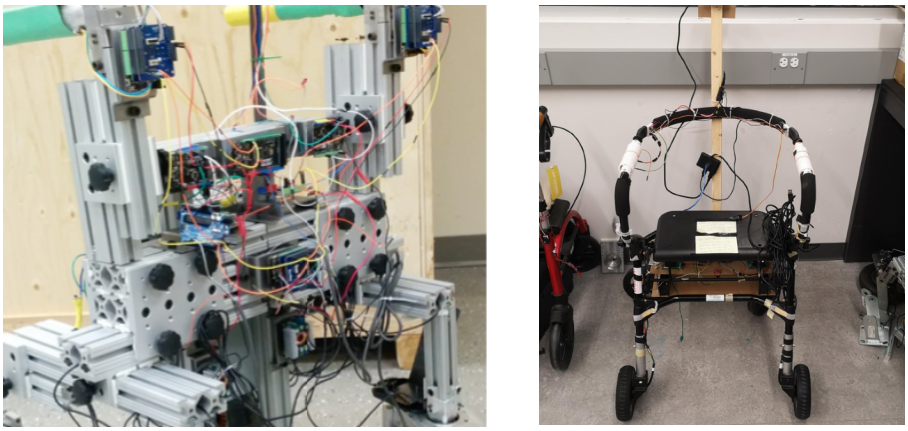


Figure 1: Old platform

Rather than designing from scratch, which can both time consuming and expensive, retrofitting required sensors into an existing off the shelf rollator would be be easier and more representative of the current products. The rollator selected to be retrofitted was the Piper Trail by Evolution weighing 13lbs and having a weight capacity of 275lbs. The rest of this chapter describing the design of the Waterloo Robotic Rollator (WATRR) is broken down into hardware and software implementation.

Table 1: Board comparison

	Jetson Nano	**Jetson TX2**	**RPi 3**	**RPi 4**
Cost	$76	$403	$48	$94 $
Memory	4GB	4GB	1GB	8GB
GPU	128 Cores	256 Cores	N/A	N/A
Power	10W	15W	5W	6.5W
Camera Compatibility	6	6	1	1

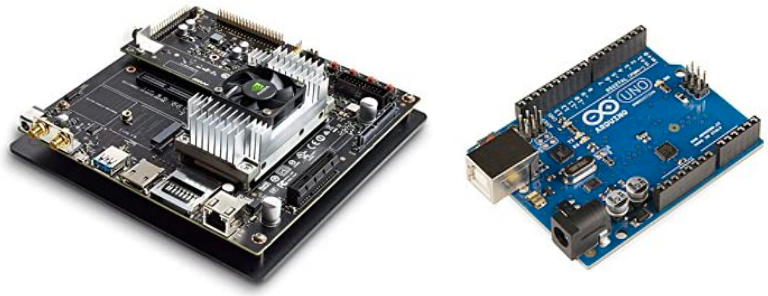
2.1 Hardware

2.1.1 Board Selection

One of the most important aspects of any robot are the microprocessor boards selected. Machine learning modules used in our systems have high computational requirements. The most common boards used for mobile robots are the Rasberry Pi series, the NVIDIA Jetson series, or single board PCs as shown in Table 1. The Jetson series was selected for our robot due to its high computational capabilities, GPU cores to facilitate computer vision, compatibility with existing electronics, and a very active developer community. While the Jetson Nano is sufficient computationally for our needs, a TX2 (seen in Figure 2(a)) was selected for research purposes, as it will allow other computationally heavy projects.

Various Arduino Uno units were selected for sensor integration and low-level control modules. Arduino is one of the most popular open source electronics platform. The Uno board (Figure 2(b)) is based on the ATmega328 chip with 14 digital input/output pins (2 hardware interrupt pins, 6 PWM pins), 6 Analog inputs, flash memory of 32KB, 2KB of SRAM, 1KB of EEPROM and a clock speed of 16MHz. It also supports UART, I2C, and SPI communication protocols. Due to its computational and memory limitations, it does not work for computationally heavy algorithms. However, the Uno is well-suited to interact with low level sensors, such as encoders and IMUs. Arduino provides an IDE which can be used to write code on a 'sketch'. The sketch can then be uploaded to the board using a USB. For code, Arduino uses a simplified version of C++ language. The Arduino community has extensive open source libraries for a lot of the commonly used sensors.

The system uses three different Arduinos subsystems: 1) Arduino Smart Handles, 2) Arduino Sensors and 3) Arduino Motor Controller. All are connected to the Jetson TX2



(a) (b)

Figure 2: (a) Nvidia Jetson TX2 (b) Arduino Uno

using USB.

2.1.2 Motor Selection and Integration

Another requirement when designing the robot is understanding how much torque is to be applied to the robot wheels to enable control. To define the torque requirements, we first considered the use of the rollator and the typical values of different parameters that affect this requirement, including:

Rollator Effective Mass: Various studies have been conducted to study the different loading cases exerted on typical rollators. In a study by Costamagna Et. al [21], experimental tests yielded maximum vertical loading exerted on the rollator at 37% of body weight. A study by Paulo Et. al [22] reported typical vertical loading of $30 - 35\%$ of a user's body weight. For a safety margin, it was decided that the robot should be able to generate sufficient torque to support 50% vertical loading of an individual. Since the selected rollator is rated for a 250 lbs (113 kg) individual, this mass was used as the requirement. Finally, we also need to take into account the mass of the rollator, which for our rollator was 15 lbs/6.8 kg.

Velocity: In the study by Costamagna Et. al [21], they report a maximum velocity of 0.76 m/s. In our walking studies, similar velocities were observed ($0.5 - 0.8$ m/s). To include a safety margin, we set our requirement at 1.0 m/s.

Acceleration: From our walking experimental study, it was seen that the user reaches their maximum velocity in about 2-3 seconds. Assuming starting from rest, a 2 second

Table 2: Rollator Parameters

Parameter	Value
Vertical loading (body weight %)	50%
User maximum mass	$113kg$
Rollator mass	$6.8kg$
User maximum velocity	$1m/s$
User ramp time	2 secs
User maximum acceleration	$0.5\frac{m}{s^2}$
Rollator wheel diameter	$6-8in$
Rollator wheel coefficient of friction	0.2
Ramp slope	5 deg

ramp and a maximum velocity of $1.0\frac{m}{s}$ equates to an acceleration of 0.5 $\frac{m}{s^2}$.

Wheel Characteristics: The wheel diameter and rolling resistance also plays an integral role in the torque requirement. While exploring motors for our robot, the wheel diameters considered were between $6-8\,in$, which is standard in many rollators. The rolling resistance coefficient was set at 0.2 which is one of the higher values reported in [23].

Ramp Incline: Extra torque is required going up ramps as the force of gravity is working against the rollator. The maximum allowable ramp slope is a $1:12$ ratio which equates to approximately 5 deg.

To summarize, the parameters set can be seen in Table 2. The torque required can be calculated using the dynamics of the system shown in Figure 3a. F_{rr} and F_f are the wheels rolling resistance and frictional forces, T is the torque applied by the motor, N is the normal force and mg is the gravitational force. Using the values from Table 2, we calculate our torque requirement to be 11.84 $\frac{N}{m}$. As we have 2 motors that will be used, the actual maximum torque requirement is half of this (5.92 $\frac{N}{m}$) at maximum operating condition. The above calculation is using the assumption that there is no wheel slip and the effect of the wheel accelerations are negligible and can be ignored. It is also assumed that the load is equally distributed among the 4 wheels, there is no user input force longitudinally and the torque is provided only by the rear wheels.

A wheel hub motor (China Drive Systems) was selected due to its simplicity and ability to provide required torque without the need of a gearing system (Figure 3b). This motor

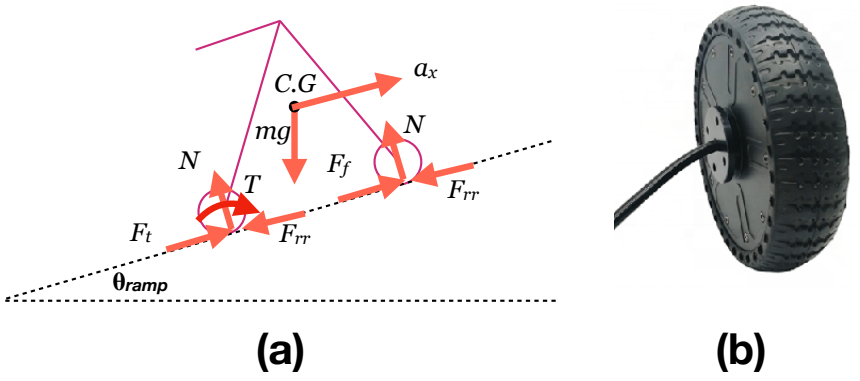


Figure 3: a) Force diagram b) Hub motor

comes with Hall effect sensors and an encoder (AEDR-8300) with a resolution of 4.35 pulses per degree. The motor has an operating voltage of 24V and a maximum torque of 13 $\frac{N}{m}$. The rated load is 5 $\frac{N}{m}$.

A motor driver board was selected to match the actuators for simplicity and safety circuitry to prevent damaging on-board electronics. EM-316 Brushless Motor Driver by Electromen, seen in Figure 4 was selected due to its features and compatibility. This driver has the capability be used as a torque and speed controller, and also has key safety features such as overvoltage, undervoltage and temperature protections. With embedded Hall effect sensors, open- and closed-loop control approaches may be considered. Controller inputs are digital signals for start/stop and direction control operations, and PWM signals are used for speed control.

A motor control circuit diagram is shown in Figure 5. Two Arduino Unos are used to control the motors, 1 for sensing and 1 for control. The 2 encoders are connected to the sensing Arduino. Due to high sampling frequency requirements, the encoder A signals are connected to digital pins 2 and 3 on the Uno which have hardware interrupts enabled and encoder B signals are connected to pins 4 and 5 which are on software interrupts. As the motor speed control requires a PWM signal, they are connected to digital pins 5 and 6 which output PWM signals at 980Hz.

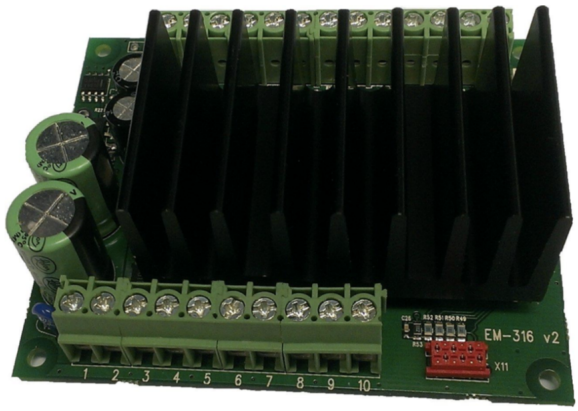


Figure 4: Motor controller [1]

2.1.3 Camera Selection

For the human state estimation module which will be discussed in later chapters, a camera with RGB and depth data, and can run at least 10Hz, is required. The camera alternatives considered can be seen in Figure 6. Important criteria of the camera are resolution, frame rate, depth range, field of view (FOV) and cost.

While the Kinect is the cheapest and one of the most used options, its minimum depth of 0.8m is not sufficient for our purpose as the human range from our rollator is typically from $0.3 - 1m$. Intel Realsense D435i, seen in Figure 8 was chosen due to its lower cost and minimum depth range of $0.1m$, making it a more suitable candidate for our purpose. While not as expensive as the Kinect community, plenty of open source libraries and documentation available for the RealSense device.

2.1.4 Sensors

The other sensors integrated in our platform include:

- **Inertial Measurement Unit**: Used to measure the angular velocity and acceleration, a LSM9DS1 IMU breakout by Sparkfun was integrated into the WATRR. Measuring acceleration, angular rotation and magnetic force in x, y and z directions, this unit interfaced with Arduino boards.

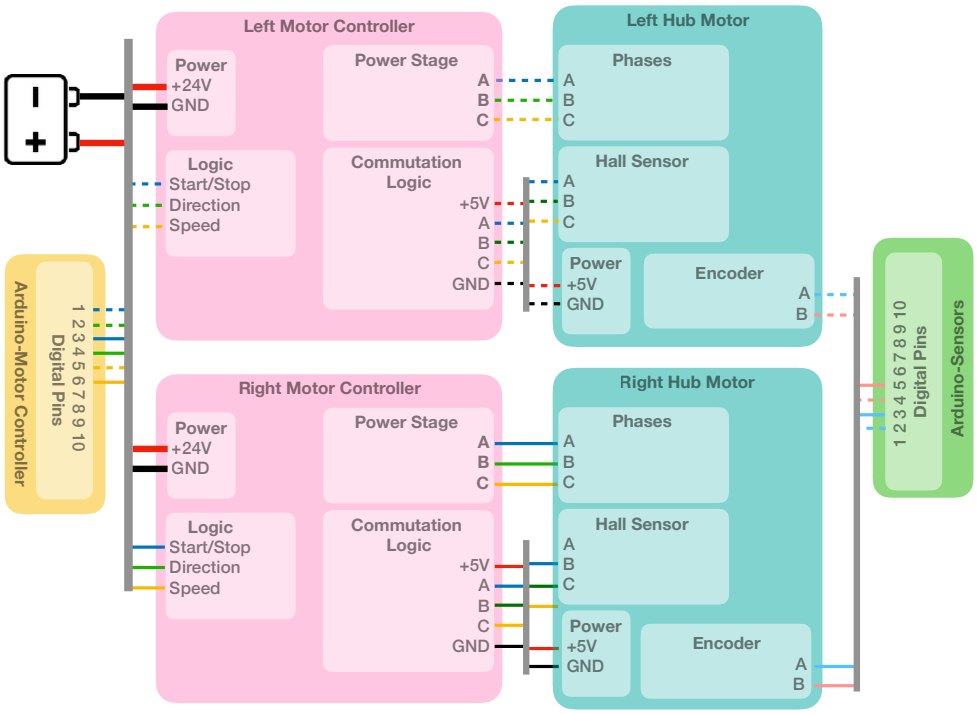


Figure 5: Motor circuit

- **Force Sensitive Resistors**: For one of our future projects, we are currently exploring a force sensing handlebar. For the handlebar, various alternatives can be used to measure the interaction forces between the human and robot. Common methods include 6 degree of freedom (DOF) load cells, strain gauges, and force sensitive resistors (FSRs). While 6 DOF load cells provide excellent measurements, their expense and bulk are clear drawbacks considering our objective. Strain gauges can be very accurate, but require high-precision mounting capabilities. While FSRs provide less accurate measurements, their capacity and ease of integration were sufficient for the purpose of capturing coarse interaction forces. Considering the primary purpose is to estimate human intent, FSRs were selected. Circular FSRs with a diameter of 0.2" by Interlink with a sensing range of $0.2 - 20N$ were selected. The maximum reported resistance change during testing (10 million actuations, $25°C$, $85°C$ was less than 10%).

Camera	Resolution	Frame Rate (1080)	Depth Range	FOV	Features	Cost
Kinect	1080	RGB-30 fps Depth- 30 fps	0.8-4.0m	70° (H) 40° (V) -° (D)	-	$139 USD
Intel Realsense D35i	1080	RGB- 30 fps Depth- 90 fps	0.1-10m	69.4° (H) 42.5° (V) 77° (D)	Built in IMU	$179 USD
ZED	720 1080 2.2K	RGB- 30 fps Depth- 100 fps	0.3-25m	90° (H) 60° (V) 100° (D)	Built in IMU	$349 USD

Figure 6: Camera comparison

Figure 7: Realsense camera [2]

- **Capacitive Sensor**: The smart handle also uses capactive sensors to check if the user's hands are on the handle before movement. For the capacitive sensor, both AT42QT1011 by Sparkfun and CAP1188 by Adafruit were tested. The CAP1188 board comes with 8 sensing lines, however it is nosier. The AT42QT1011 has only 1 line but has a much more reliable measurement, thus it was selected for our platform.

Figure 8: Platform sensors: a) IMU b) FSR c) Capacitive

2.2 Software Architecture

One of the most difficult challenges implementing robots is maintaining code complexity management, especially when the scale and scope of the robot grows [24]. Development of software architectures from scratch requires a breadth of expertise and typically well beyond a single researcher's capabilities, thus making it difficult to prototype robots quickly [24]. Quigley Et. al from Stanford University and Willow Garage have developed 'Robot Operating System (ROS)', a robotics software framework which is easy to use, free, open source and featuring a highly generalized architecture.

The main advantages of ROS are:

- **Peer to Peer Architecture**: Most robots consists of multiple processes, often running on multiple hosts which can make communication problematic. A peer-to-peer network simplifies this vastly [24].
- **Multi-Lingual**: All developers have different preferences of languages they like to use for development. ROS is language neutral and supports various languages including C++ and python [24].
- **Thin**: While many robotics software projects contain potentially re-useable drivers and algorithms, it is often very difficult to extract the functionality of the code due to the dependence on middleware. ROS addresses this issue by encouraging standalone code without dependency on ROS itself. ROS offers catkin, its own build system which enables this 'thin' execution [24].
- **Free and Open Source**: Probably the biggest advantage of all is that ROS is both free and open source. This has opened many rapid advancements in ROS development to the entire community, including support across a wide range of platforms and electronics [24].

A system built on ROS contains **nodes**, which is essentially a software module or process. Nodes are able to communicate with each other through **messages**, which is a data structure including integers, floats, text, arrays, etc. One can create their own message types as required by the system. Each node has the capability to both subscribe or publish messages to multiple **topics**, which is essentially a data of interest. Different nodes can publish to the same topics and are not aware of other publishers and subscribers of the topic. ROS also offers synchronous transactions through the use of **services**, which have a defined request and response message type.

WatRR's ROS architecture is shown in Figure 9. The arrows show the direction of data transfer. The black arrows represent the node-topic publishing while the gray ones represent the subscribers.

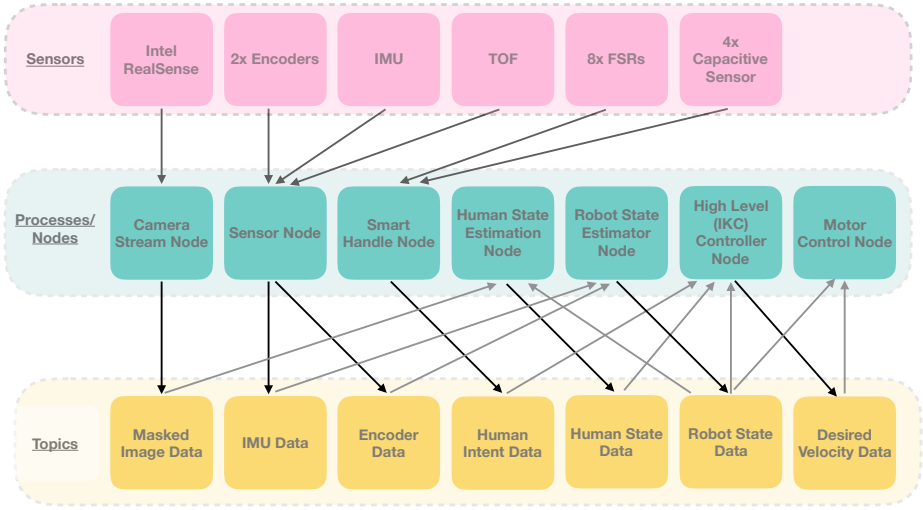


Figure 9: ROS architecture

Camera Stream Node: The purpose of this node is to initialize the camera, stream the data from the Intel RealSense, process it and publish the data to the 'Masked Image Data' topic. This node is coded in Python 3 and processed on the Jetson TX2. For the processing, the RGB and depth data is aligned using the Intel RealSense SDK. The SDK is also able to provide the depth data in X, Y, Z coordinates separately which is used to mask the image before publishing it at 10Hz.

Sensor Node: The purpose of this node is to interface with the IMU, encoders and the time of flight sensor, process the data and publish it to the 'IMU Data' and 'Encoder Data' topics. This node is processed on the 'Arduino-Sensor' board and written in Arduino code.

It uses the built in libraries for IMU. As for the encoder, we use 4x encoding. Each of the lines from the encoder can have a HIGH or LOW value and depending on the current and last state, the encoder counter is incremented or decremented, which is used to calculate the angle of rotation before publishing at 100Hz.

Smart Handle Node: The purpose of this node is to interface with the handle sensors (FSRs and capacitive sensors) and estimate the intent based on the values. This node is written in Arduino code and processed on the 'Arduino-Smart Handles' board. It then publishes the intent to the 'Human Intent Data' topic. As this module is still under work, it is not fully used in the control systems currently and is not covered in this thesis.

Human State Estimation Node: The purpose of this node is to use the masked image to predict shoulder locations, and use shoulder information in conjunction with depth and robot data to estimate the human state. This node is coded in Python 3 and processed on the Jetson TX2. The shoulder estimator is trained using Keras and TensorFlow, both open source machine learning platforms, and optimized using TensorRT to improve run time. The predictions are made using the optimized weights and TensorFlow. Finally, once the human state is computed, it is published to the 'Human State Data' topic at 10 Hz.

Robot State Estimation Node: The purpose of this node is to calculate robot and wheel velocities and the robot position. This node is written in Python 3 and processed on the Jetson TX2. It uses the encoder and IMU data to estimate the state and publishes it to the 'Robot State Data' topic at 100Hz.

High Level (IKC) Controller Node: The purpose of this node is to generate the desired velocities of the robot. This node carries out the computations seen in Section 3. It is subscribed to the robot state, human state and human intent data topics. These are all then used in the IKC Controller and published to the 'Desired Velocity Data' topic.

Motor Control Node: The purpose of this node is to process the velocity tracking controller and output the desired voltages to the motor driver. This node is subscribed to the 'Desired Velocity Data' and 'Robot State Data' topics, written in Arduino code and processed on the 'Arduino- Motor Control' board. It takes the desired velocity data and performs the feedback calculation. It also tracks the wheel velocities and processes the wheel PID controller. Using those signals, a PWM signal is determined and sent to each wheel using digital pins 5 and 6.

3 Control Design

3.1 Background

As mentioned earlier, a literature search yielded numerous control systems proposed for robotic rollators. The JARoW is an active robotic rollator which predicts the users intent based on shin location as an indicator of gait cycle detection using infrared sensors [12]. Depending on the shin locations, they classify the intent in 4 different states: halt, step left/right with minimal forward movement, move forward, turn left/right. This intent is then used to generate desired linear and angular velocities for on-board motor control. While this project has shown promising results, the supporting evidence is limited in sample size (3 young healthy males) and does not report how controls are adjusted to an individuals' gait pattern, an important factor to be considered when designing control systems for a heterogeneous population with sensory and/or motor impairment(s).

To ascertain human mobility intent, Morris Et al. developed a platform with a handle bar embedded with 2 force sensitive resistors measuring the force in the robot longitudinal to inform generation of the desired velocites directly [19]. There are 3 modes of operation: passive, active, and forced mode. In passive mode, the user is in complete control of the walker. The robot's function is to prevent collisions and monitor the user's position. For active mode, the robot trajectory is used as desired and the users trajectory is tracked. If the deviation between the desired and actual trajectories increases beyond a threshold, it begins to slow down until the user aligns. If the user does not align, the robot comes to a complete halt. For the forced mode, the robots trajectory is used and the user feedback is completely ignored. Using the user force feedback directly is not always favorable as it typically causes oscillations and may also lead to accidents if the user direct input is unintentional. However, the idea of using a force sensing handle bar is promising, as is automatically halting the robot when the path deviation is high. Graf proposed a control algorithm using direct user force input in the handles to calculate the desired velocities based on dynamic modelling, combined with environmental sensors to detect obstacles [25]. Once again, using direct force input can be dangerous due to unintentional or erroneous inputs. Graf's robotic rollator platform was also bulky, which would heavily restrict stair ascent/descent tasks.

Cifuentes Et al and Halit [26, 27] both proposed using an Inverse Kinematic Controller (IKC) to control the robot. The goal of this controller is to control the distance and the heading angle of the rollator in relationship to the user, such that they are in line and a safe distance is maintained. Another benefit of this controller is the indirect input of the user, as it employs the user's state to control the robot. Cifuentues Et al. also

proposed combining the IKC with direct inputs to achieve better control of the rollator, demonstrating improvements in yaw control. While both of the aforementioned papers present great ideas, evidence to support real-world implementation remains lacking. While Halit Et al. demonstrated feasibility and improved performance with both PID and sliding mode controller, their study was solely in simulation and did not present any experimental data. While Cifentes Et al. developed a physical platform, their proposed solution requires sensors to be donned by the user (e.g IMU), includes several expensive components (3DOF force sensors), and is physically bulky.

This thesis builds on the aforementioned research base, and employs learning-based techniques from these projects to improve control methods using a hybrid controller approach, and novel estimation methods.

3.2 Controller design for the robotic rollator

The goal of the proposed platform is to control the distance and orientation of the WATRR (or 'Rollator') with respect to the human (or 'user'). This control scheme was chosen for 2 main reasons: 1) In Riel's study for falls while using a rollator, they suggested that improper use of rollator leads to slips and falls, with one reason being the user not being positioned in between the handholds while pushing it [16], which can be addressed through the proposed controller, and 2) while additional weight discourages use, the energy expenditure and the apparent weight of the robot may be reduced by providing assistance [16]. To achieve an assistance controller, an appropriate human-robot kinematic model is required. The model explored in [26, 27] can be seen in Figure 10. We employ this model as a baseline and improve upon the prior (existing) control systems approach. This model will be described in greater detail here.

3.2.1 Kinematic Modeling

The rollator can be modeled as a differential drive robot with the two rear wheels being actuated and the front wheels are on casters. The rollator coordinate frame can be placed at point R, which is at the midpoint of the two wheels at the rear axis. The pose of the rollator in the global frame O, is specified by the state vector $P = \begin{bmatrix} x_r & y_r & \alpha_r \end{bmatrix}^T$ where x_r, y_r are the coordinates of point R and a_r is the orientation of the rollator w.r.t the global coordinate. The kinematic model can be written as:

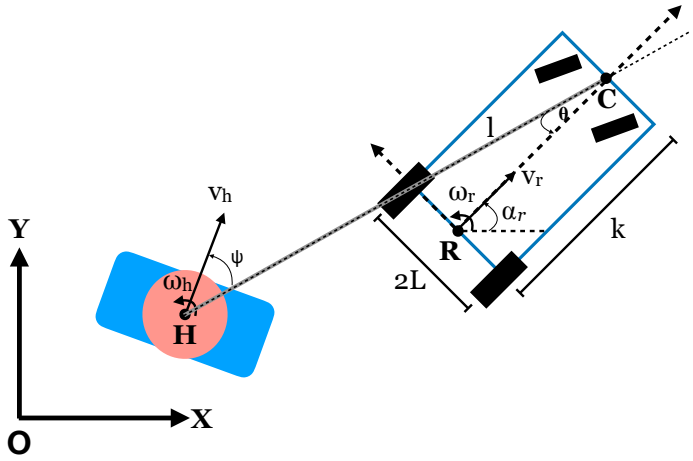


Figure 10: Kinematic model

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\alpha}_r \end{bmatrix} = \begin{bmatrix} \cos(\alpha_r) & 0 \\ \sin(\alpha_r) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} \tag{1}$$

where v_r, ω_r represent the rollator's linear and angular velocity respectively. The rollator linear and angular velocity can be related to the wheel speeds using:

$$\begin{bmatrix} v_r \\ \omega_r \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2L} & \frac{-r}{2L} \end{bmatrix} \begin{bmatrix} \dot{\theta}_R \\ \dot{\theta}_L \end{bmatrix} \tag{2}$$

where θ_R, θ_L are the rotation displacement of the right and left wheel respectively and $\dot{\theta}_R, \dot{\theta}_L$ are wheel rotational velocities. The radius of the wheel and the rollator wheelbase is given by r and $2L$, respectively.

Combining Equations 17 and 2, the forward kinematic model can be expressed as

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\alpha}_r \end{bmatrix} = \begin{bmatrix} \frac{r}{2}\cos(\alpha_r) & \frac{r}{2}\cos(\alpha_r) \\ \frac{r}{2}\sin(\alpha_r) & \frac{r}{2}\sin(\alpha_r) \\ \frac{r}{2L} & -\frac{r}{2L} \end{bmatrix} \begin{bmatrix} \dot{\theta}_R \\ \dot{\theta}_L \end{bmatrix} \tag{3}$$

The above equation holds under the assumption that there is no lateral slip, that is the lateral velocity of the rollator in the body frame is zero, and is given by:

$$-\dot{x}_r \sin(a_r) + \dot{y}_r \cos(a_r) = 0 \tag{4}$$

The main goal of our control strategy is to control the distance and orientation of the rollator with respect to the human. The kinematic model of the human-robot interaction in the polar coordinates can be seen in Figure 11. Here location H and C are the coordinates of the human center point and the front-center of the robot w.r.t the global coordinate system respectively. v_h, w_h are the human linear and angular velocities respectively. Parameter l is the distance between point C and H, θ is the angle between the longitudinal axis of the robot and the axis l and ψ is the relative distance between the human and robot heading angle.

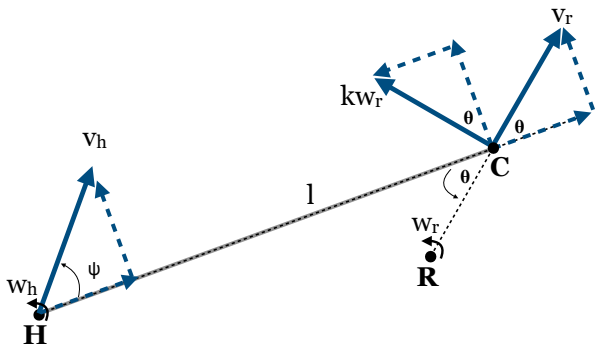


Figure 11: Trig model of human-robot kinematics

From the given figure, the relative distance-orientation equations can be found:

$$\dot{l} = v_r \cos(\theta) - k w_r \sin(\theta) - v_h \cos(\psi) \tag{5}$$

$$\dot{\psi} = \frac{-v_r \sin(\theta)}{l} \frac{-k w_r \cos(\theta)}{l} + \frac{v_h \sin(\psi)}{l} + w_h \tag{6}$$

The matrix form of this is given by:

$$\begin{bmatrix} \dot{l} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -k \sin(\theta) \\ \frac{-\sin(\theta)}{l} & \frac{-k \cos(\theta)}{l} \end{bmatrix} \begin{bmatrix} v_r \\ w_r \end{bmatrix} + \begin{bmatrix} -v_h \cos(\psi) \\ \frac{v_h \sin(\psi)}{l} + w_h \end{bmatrix} \tag{7}$$

Using the above equation the inverse kinematics model is:

$$\begin{bmatrix} v_r \\ w_r \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -l\sin(\theta) \\ \frac{-\sin(\theta)}{k} & \frac{-l\cos(\theta)}{k} \end{bmatrix} \begin{bmatrix} \dot{l} + v_h\cos(\psi) \\ \dot{\psi} - w_h - \frac{v_h\sin(\psi)}{l} \end{bmatrix} \tag{8}$$

3.2.2 Human Trajectory Simulation

For human trajectory generation, the human path was modeled similar to a robot and is given as follows:

$$\begin{bmatrix} x_h \\ y_h \\ \alpha_h \end{bmatrix}_{t+1} = \begin{bmatrix} x_h \\ y_h \\ \alpha_h \end{bmatrix}_t + \begin{bmatrix} \cos(\alpha_h) & 0 \\ \sin(\alpha_h) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_h \\ \omega_h \end{bmatrix} \Delta t + \begin{bmatrix} \sin(\alpha_h) \\ \cos(\alpha_h) \\ 0 \end{bmatrix} \begin{bmatrix} v_d \end{bmatrix} \Delta t \tag{9}$$

where $\begin{bmatrix} x_h & y_h & \alpha_h \end{bmatrix}^T$ is the human position state vector in the global coordinates, v_h is the human linear velocity and w_h is the human angular velocity. While walking, humans also tend to move laterally with each step and is modeled as a disturbance velocity, given by v_d. Here, the inputs to the system are v_h and w_h.

Furthermore, it was observed through data analysis that the human velocity typically follows a cyclic pattern which corresponds to steps. Therefore, the input velocity was modeled as follows:

$$v_{h,t} = v_{amp}\sin(\omega t) + v_{avg}. \tag{10}$$

The shoulders of the human also swings back and forth while walking as the user does not always walk straight. From the persons center, the shoulder locations were found using:

$$\begin{bmatrix} x_{ls} \\ y_{ls} \\ x_{ls} \\ y_{ls} \end{bmatrix} = \begin{bmatrix} x_h - \frac{b_w \sin a_{r,s}}{2} \\ y_h + \frac{b_w \cos a_{r,s}}{2} \\ x_h + \frac{b_w \sin a_{r,s}}{2} \\ x_h - \frac{b_w \cos a_{r,s}}{2} \end{bmatrix} \tag{11}$$

where $a_{r,s} = a_r + a_{r,amp}\sin(wt)$. Here b_w is the user shoulder width, $a_{r,amp}$ is the amplitude of shoulder angular rotation, w is the step period $(2\pi f)$.

To validate the walking model is representative of reality, multiple experimental trials (3x straight, 3x left turn, 3x right turn) were conducted. One healthy young female aged

23, was asked to walk with a manual (passive) rollator and perform 3 different walking trials: 1) walk straight, 2) walk straight then turn right, 3) walk straight and then turn left. Data was collected using Vicon motion capture with markers placed on the left shoulder, right shoulder and head of the user and on the camera, left handle and right handle on the rollator. All data were captured and post processed using MATLAB (v2019a). The results can be seen in Figures 12, 13, 14 where RS path and LS path denote the position of the right shoulder and left shoulder respectively. There are four important observations that can be made from these figures: 1) the linear velocity of the person is not constant, typically oscillating in synchrony with their steps, 2) the shoulders of the person swing back and forth in the XY plane, as seen in the shoulder paths and more evident in the human angular velocity, where we observe an oscillatory behaviour pattern, 3) humans typically do not walk completely straight, but includes some lateral motion, as seen in the shoulder paths, 4) while the shoulders swing back and forth, the human typically controls the rollator such that it follows a smooth path. The aforementioned variance in the shoulder paths has been modeled as a disturbance in the trajectory model.

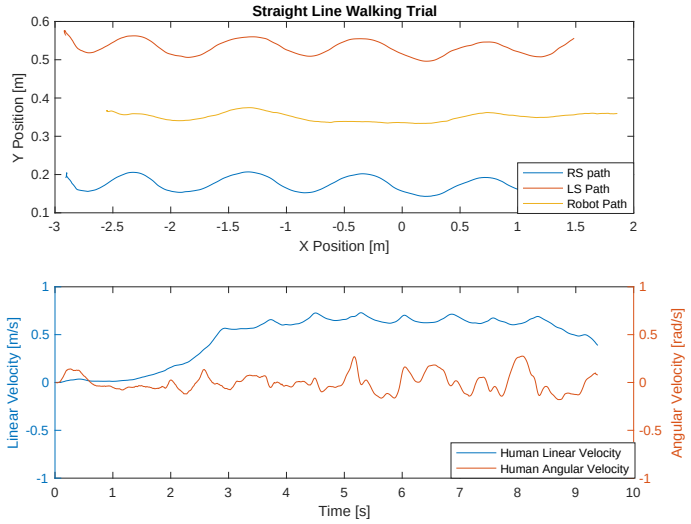


Figure 12: Experimental walking trial: straight path

3.2.3 Inverse Kinematic Controller

From the above kinematic model in Eqn 8, an inverse kinematic controller (IKC) was developed using the following control law proposed in [27].

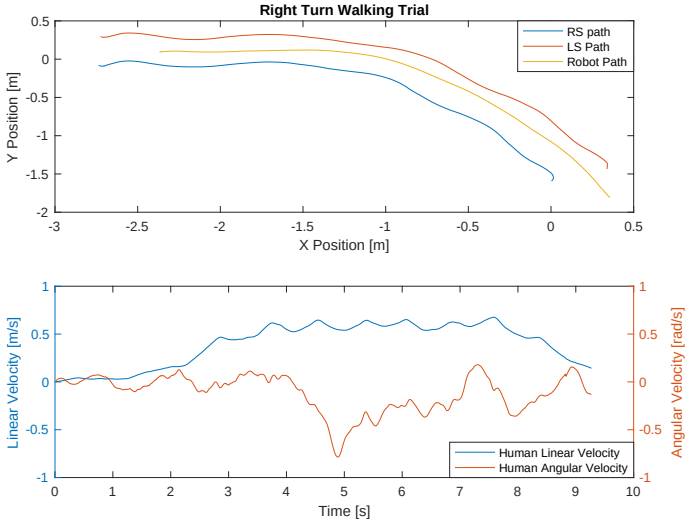


Figure 13: Experimental walking trial: right turn

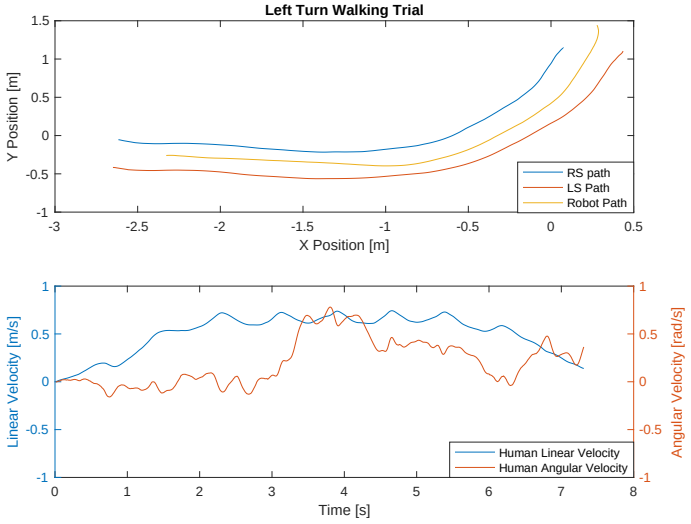


Figure 14: Experimental walking trial: left turn

$$\begin{bmatrix} v_r \\ w_r \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -l\sin(\theta) \\ \frac{-\sin(\theta)}{k} & \frac{-l\cos(\theta)}{k} \end{bmatrix} \begin{bmatrix} -k_{p,l}e_l + v_h\cos(\psi) \\ -k_{p,\psi}e_\psi - w_h - \frac{v_h\sin(\psi)}{l} \end{bmatrix} \tag{12}$$

where $e_l = l - l_d$ and $e_\psi = \psi - \psi_d$. Here l_d and ψ_d are the desired distance and relative heading angle.

$$\begin{bmatrix} \dot{e}_l \\ \dot{e}_\psi \end{bmatrix} = \begin{bmatrix} -k_{p,l} e_l \\ -k_{p,\psi} e_\psi \end{bmatrix} \tag{13}$$

Given $k_{p,l}, k_{p,\psi} > 0$, this system is Hurwitz and asymptotically stable.

3.3 Simulation study

To validate the kinematic model, a simulation study was conducted replicating prior work [27] using newly acquired experimental data. Three different test cases were simulated which represented the situations of interest. Three paths were simulated: 1) straight path trajectory, 2) turning trajectory, 3) difference in heading angle between the human and robot. For all simulations, the following parameters were used:

Parameter	Value	Unit
Controller Settings		
l_{des}	0.6	m
a_{des}	0	rad
$k_{p,l}$	0.5	$\frac{1}{s}$
$k_{p,\psi}$	1.5	$\frac{1}{s}$
Simulation Settings		
$v_{h,amp}$	0.1	$\frac{m}{s}$
$v_{h,avg}$	0.5	$\frac{m}{s}$
v_d	0.1	$\frac{m}{s}$
$a_{r,s}$	0.1	rad
f	0.5	$\frac{steps}{s}$
b_h	0.4	m
ts	0.05	sec

Test Case I: Straight Path Trajectory

For the straight path trajectory, the initial state vectors were set to $X_h = [0, 0, 0]$, $X_r = [1.0, 0, 0]$.

From the results in Figure 15, it can be seen that the distance error is asymptotically driven to 0 over time. Oscillatory behaviour in the distance behaviour was observed, driven by the cyclic behaviour of the human walking velocity profile. Furthermore, it is important to observe that the robot path also has an oscillatory behaviour, attributable to the swinging of the shoulders.

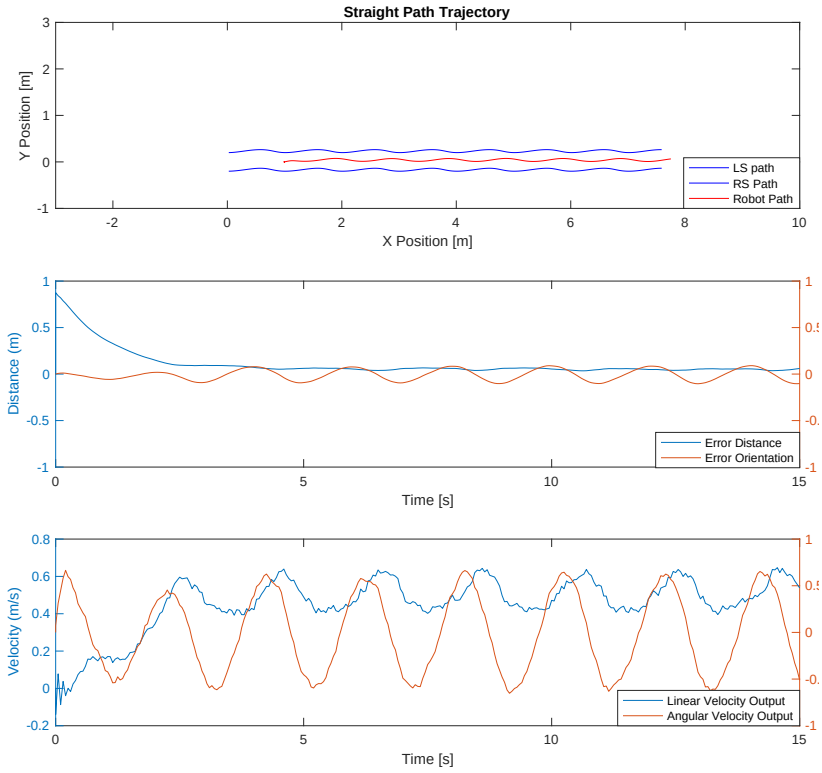


Figure 15: Simulation result: straight path

Test Case II: Left Turn Trajectory

To simulate turns with the robot, a straight-line walking for 7.5 seconds at an average velocity, followed by turning for 7.5 seconds at a rate of $w_h = 0.25rad$ was used. The initial state vectors were once again set to $X_h = [0, 0, 0]$, $X_r = [1.0, 0, 0]$.

Similar to the straight path trajectory, Figure 16 illustrates the distance and orientation error was asymptotically driven to zero with some oscillatory behavior.

Test Case III: Difference in heading angle

For this simulation, the robot heading and the human heading angle were set to different directions at the start of the simulation. The purpose was to ensure that the robot corrects itself to match the heading of the human when there is a large difference. The person was set to walk straight for 15 seconds for this simulation. The initial state vectors were set to $X_h = [0, 0, 0]$, $X_r = [1.0, 0, 0.785]$ and the results are shown in Figure 17. The robot is able

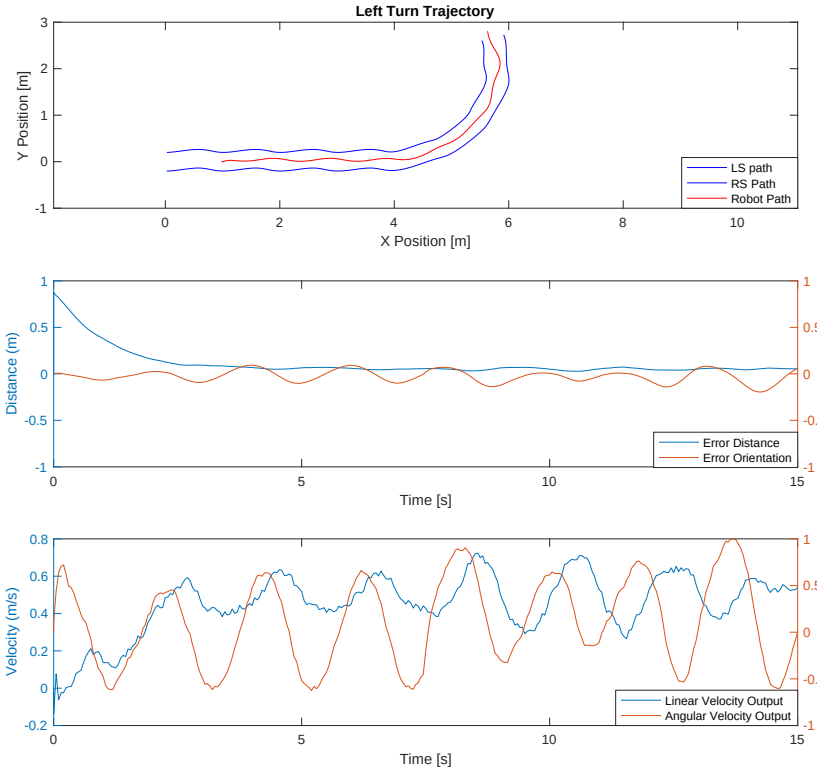


Figure 16: Simulation result: left turn

to correct its heading error fairly quickly with a similar response to those shown above.

One of the primary observations made from the above results is the oscillation of the robot during normal gait due to the swinging of the shoulders. One of the flaws of the proposed controller is that its goal is to drive the orientation error to zero, leading to oscillatory behaviour due to controller sensitivity to swinging of the shoulder and lateral velocity of the person. When the user walks with the robot in reality, the goal should focus on a smoother trajectory (i.e., not tied to shoulder and lateral oscillations). For example, when the user is walking straight, the robot should continuously move straight along with the user and not swing back and forth with the user's shoulders. A good example of this can be seen in Figure 12 where the user performed a walking trial with **no** input from the robot. Shoulder positions have a cyclic pattern at a higher frequency than the robot path

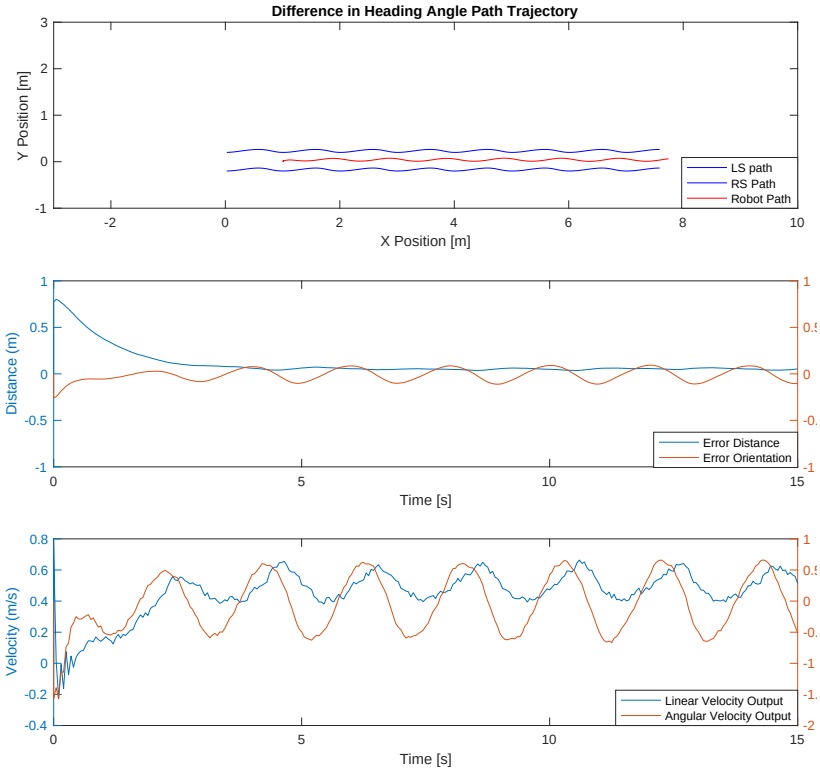


Figure 17: Simulation result: difference in heading angle

which is much smoother. The goal of the controller is to achieve similar results as unaided gait and hence, a new controller with such characteristics is desired.

3.3.1 Hybrid Control

To improve upon the previous controller, as well as address more cases, we propose a modification to the IKC. The goal of this extension is to have a smoother trajectory as well as implement the following features to increase safety by continuously maintaining a stable base of support.

- Users' tendency to drag the rollator backwards with them while falling has been

reported. Due to this, the controller has been advanced such that the IKC only outputs positive velocities, and is never driven backwards towards the user.

- A smoother trajectory with attenuated high frequency oscillations due to the swinging of the shoulders is desired.
- When the robot is far away or has a very large difference in heading angle compared to the user, it poses a safety concern and the robot should be braked (or moved) immediately to realign the robot

To address the above issues, a hybrid controller is proposed. In this hybrid controller, we propose three states: 1) Passive State 2) Active State 3) Brake State. Figure 18 shows the different zones where the different modes operate. Here, green zones are active areas, blue zones are passive areas and red zones are brake areas.

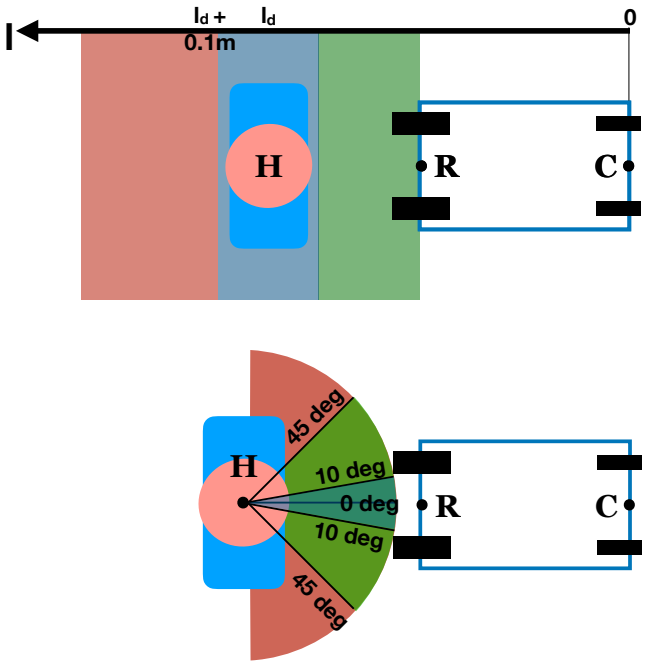


Figure 18: Hybrid controller zones

Passive State: In the passive state, the robot continues to sense and provides little assistance to the user. The IKC controller's goal is to maintain a desired distance and

orientation. Considering human gait with a rollator is irregular and an exact distance and orientation is not strictly maintained while normal walking, the prior IKC implementation [27] causes the rollator to oscillate during straight walking due to the sensitivity in distance and orientation error. To reduce this oscillation and generate a smoother trajectory, we propose a passive zone as seen in Figure 18, where the effect of the error on the output velocity is minimal as the user is still within a safe zone. In this passive state, while the controller still outputs wheel velocities, the controller does not significantly weight the distance error, and hence the $k_{p,l}$ is set to 0.5 while the distance error is within 0.1m. The same approach is applied for angular control. While output wheel velocities are still computed, the contribution of e_ψ is reduced by setting $k_{p,\psi}$ to 0.5. As the robot is also sensitive to human angular velocity, $w_h = 0$ was set to prevent yaw motion while the angle is in this zone.

Active State: In the active state, an IKC with varying gain schedule to control the robot is proposed. When the error is low, an aggressive response is not desired. As the error increases, it poses a safety concern and the controller is asked to drive the system to stability more quickly. The gain schedule proposed is as follows:

$$k_{p,l} = \begin{cases} \max(\min(5e_l, 2.5), 0.5) & e_l > 0 \\ \max(\min(-5e_l, 2.5), 0.5) & e_l < 0 \end{cases} \tag{14}$$

$$k_{p,\psi} = \begin{cases} 0.5 & |e_\psi| < 0.17rad \\ \min(5|e_\psi|, 2.5) & |e_\psi| > 0.17rad \end{cases} \tag{15}$$

The effect on error with different $k_{p,l}$ values can be seen in Figure 19. As expected, as the gain increases, the response is more aggressive with a quicker response time. However, that comes at a cost on the overshoot of the system. To deal with the overshoot, a controller with a varying gain schedule is proposed, where the gain increases with the error. During the passive state, the gain is set to 0.5 to dampen the error correction. The values were selected experimentally. Similar to Ziegler Nichols method [28], we increased our gains until the system started oscillating unstably, which for our case was at a gain of 5.0. We then selected a gain of half of that for a maximum gain of 2.5 for the system. The minimum gain was chosen to be 0.5 based on the minimal error and oscillations shown in Figure 19.

Brake State: In the brake state, we propose a PID controller for wheel position. When the error becomes very large, it poses a safety concern for the user. For example, a documented reason for falls is the rollator rolling away from the user [15]. In such high

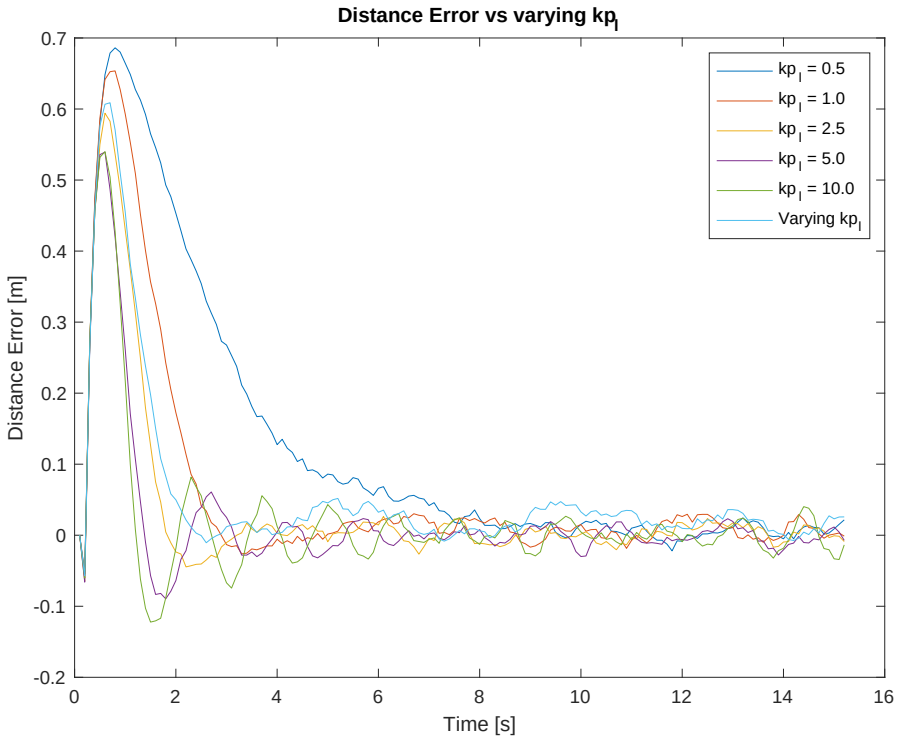


Figure 19: Distance error vs gain

risk situations, the robot should be halted until the user is realigned with the rollator. The equation for the PID controller can be given by:

$$v_{out,i} = k_{p,\theta} e_{\theta,i} + k_{d,\theta} \dot{e}_{\theta,i} \tag{16}$$

Here $v_{out,i}$ represents the output wheel signal. $k_{p,\theta}$ and $k_{d,\theta}$ are the proportional and integral gain respectively. This braking state is mainly for safety precautions.

The simulation results from the above controller can be found in Figures 20-21. As seen from the results, the robot path for the hybrid controller is smoother than the pure IKC implementation. As desired, the hybrid controller is less sensitive to the swinging of the shoulder, indicated by the diminished amplitude of the oscillations.

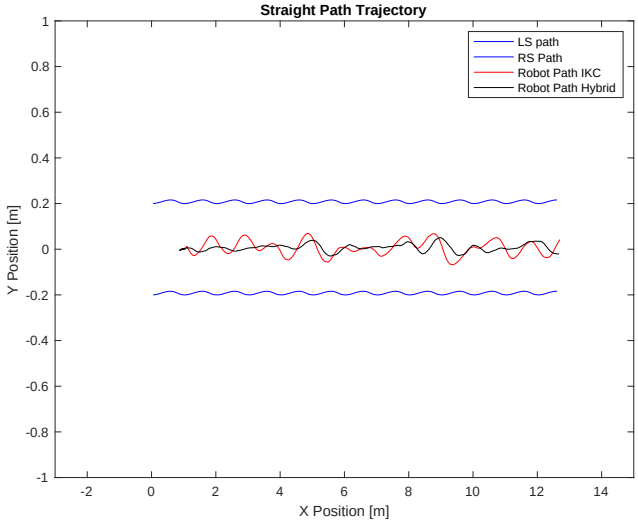


Figure 20: Hybrid controller simulation result: straight path

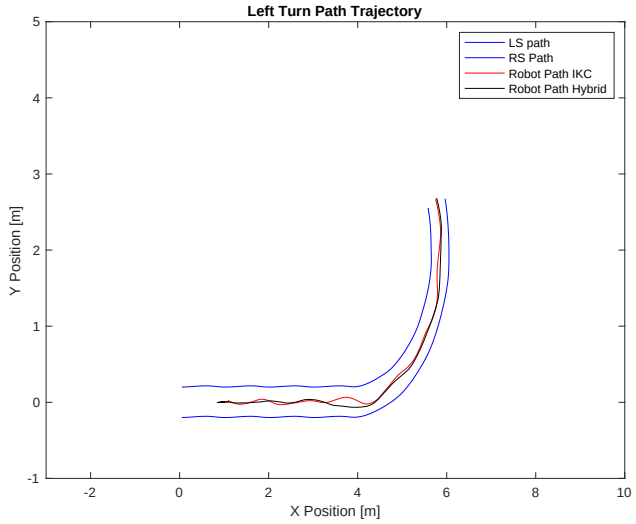


Figure 21: Hybrid controller simulation result: left turn

3.4 Velocity Tracking Controller

The above simulations assumes 'perfect velocity tracking' which means that the output velocity of the rollator is the same as desired velocity. Many current works use this assumption which have poor steady state and transient responses due to unknown distur-

bances and modelling errors. [29]. Some sources of error can be wheel alignment, incorrect effective wheel radius, and/or incorrect model parameters among other reasons. While we can output a desired velocity, the actual output is not as desired due to these external disturbances. Similar to [29], we propose a controller structure that uses feed-forward and feedback control elements. The feed-forward velocity is obtained from the IKC in the previous section. The feedback controller is designed using a PID controller. The proposed method can be seen in Figure 22.

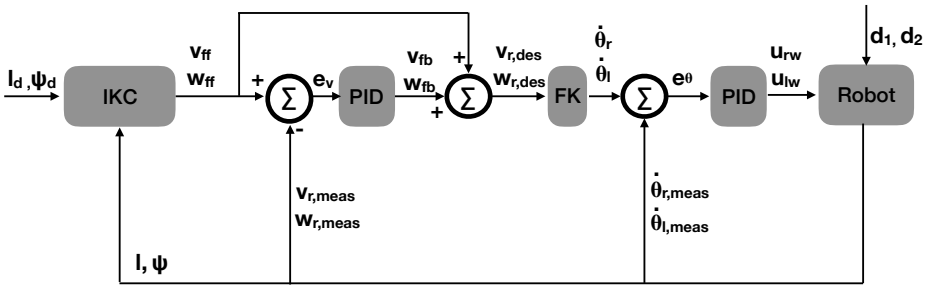


Figure 22: Proposed controller

The kinematic model of the rollator in the presence of unknown model uncertainties and disturbances can be given by:

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\alpha}_r \end{bmatrix} = \begin{bmatrix} \cos(\alpha_r) & 0 \\ \sin(\alpha_r) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} + \begin{bmatrix} \cos(\alpha_r) & 0 \\ \sin(\alpha_r) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \tag{17}$$

where d_1 and d_2 are the linear and angular disturbances due to unmodeled dynamics, structural variations and disturbances. It is assumed that these are bounded, where $|d_1| < \delta_1$ and $|d_2| < \delta_2$.

As mentioned earlier, the desired output velocity for the controller can be obtained using a feed-forward and a feedback term. The feed-forward is found using the hybrid controller as discussed in the previous section and the feedback is found using a PID controller. The output at each time step is:

$$\begin{bmatrix} v_{r,des} \\ w_{r,des} \end{bmatrix} = \begin{bmatrix} v_{ff} \\ w_{ff} \end{bmatrix} + \begin{bmatrix} v_{fb} \\ w_{fb} \end{bmatrix} \tag{18}$$

where,

$$\begin{bmatrix} v_{ff} \\ w_{ff} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -l\sin(\theta) \\ \frac{-\sin(\theta)}{k} & \frac{-l\cos(\theta)}{k} \end{bmatrix} \begin{bmatrix} -k_l e_l + v_h \cos(\psi) \\ -k_\psi e_\psi - w_h - \frac{v_h \sin(\psi)}{l} \end{bmatrix} \tag{19}$$

and

$$\begin{bmatrix} v_{fb} \\ w_{fb} \end{bmatrix} = \begin{bmatrix} k_{p,v} e_v + k_{i,v} \int_0^t e_v dt \\ k_{p,w} e_w + k_{i,w} \int_0^t e_w dt \end{bmatrix} \tag{20}$$

Here $e_v = v_{ff} - v_{r,meas}$ and $e_w = w_{ff} - w_{r,meas}$. We can then find the wheel velocities using the forward kinematics equation:

$$\begin{bmatrix} \dot{\theta}_r \\ \dot{\theta}_l \end{bmatrix} = \begin{bmatrix} 1 & K \\ 1 & -K \end{bmatrix} \begin{bmatrix} v_{r,des} \\ w_{r,des} \end{bmatrix} \tag{21}$$

Finally, we implement a wheel level PI controller to track the wheel velocities defined as:

$$\begin{bmatrix} u_{rw} \\ u_{lw} \end{bmatrix} = \begin{bmatrix} k_{p,\dot{\theta}} e_{\dot{\theta},r} + k_{i,\dot{\theta}} \int_0^t e_{\dot{\theta},r} dt \\ k_{p,\dot{\theta}} e_{\dot{\theta},l} + k_{i,\dot{\theta}} \int_0^t e_{\dot{\theta},l} dt \end{bmatrix} \tag{22}$$

where u_{rw} and u_{lw} are the right and left wheel voltage control signals respectively and $k_{p,\dot{\theta}}$ and $k_{i,\dot{\theta}}$ are the wheel velocity proportional and integral gains. The wheel velocity error is calculated as $e_{\dot{\theta},i} = \dot{\theta}_i - \dot{\theta}_{i,meas}$.

The simulation results from the above controller can be seen in Figures 23-24. The linear velocity disturbance was set to a constant $0.1\frac{m}{s}$ with a step up to $0.5\frac{m}{s}$ from $t = 5 - 7s$. For the angular velocity disturbance, a constant of $-0.1\frac{rad}{s}$ was used with a step up to $-0.5\frac{rad}{s}$ from $t = 10 - 12s$. $k_{p,v}$ and $k_{p,w}$ are set to 1 while $k_{i,v}$ and $k_{i,w}$ are set to 0.5.

As seen from the results, the controller is able to track the velocities in the presence of velocity disturbances which can be due to modelling errors or the physical platform parameter variations such as wheel alignment and wheel radius.

3.5 Experimental Results

The Hybrid controller above was implemented using our custom WATRR platform, shown in Figures 25-26. To validate our controller at this juncture, the tests performed were

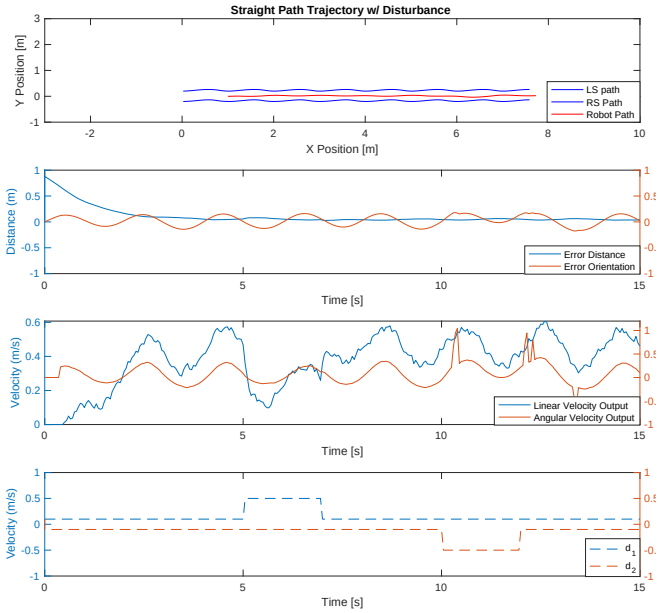


Figure 23: Simulation result: straight path w/ step disturbance from 10-12 secs

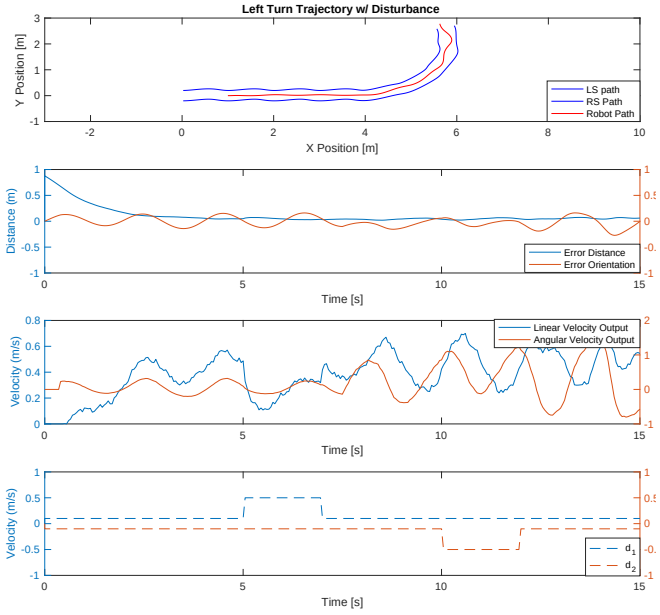


Figure 24: Simulation result: left turn w/ step disturbance from 5-7 secs

carried out completely autonomously (i.e., without human interaction). In reality, human-robot interactions need to be considered for a full evaluation.

The tests were carried out in a lab with 1 healthy male, aged 25. The floor was marked with tape and the participant was asked to follow the marked floor.

The maximum linear and angular velocity were set to $0.4\frac{m}{s}$ and $0.78\frac{rad}{s}$. The IKC gain functions selected can be found in Section 3.3.1. Based on experiments, $k_{p,v}$, $k_{p,w}$, $k_{i,v}$, $k_{i,w}$ are set to 0, 0.5, 0, 0.2 respectively.

As mentioned earlier, the main goal of our controller is to control the rollator such that it is in front of the user at a desired distance l_d and a desired orientation ψ_d as $t \to \infty$. The experimental data can be seen in Figures 25-26. From the top graphs of the results, we can see that both the distance and the orientation error decreases over time.

Another observation to be made is that there is often a large error between the desired and actual rollator velocities (both linear and angular). This is likely attributed to external disturbances, system delay, not accounting the rollator dynamics and the wheel velocities being bounded. At around the 190th time step in Figure 25 linear velocity plot, the desired output velocity drops to 0. However it can be observed that the system does not respond until around the 200th time step. This is expected since there is always some delay in the system and the controller response. This can also be due to not accounting for the system dynamics. When the desired velocity suddenly drops to zero, no torque is applied to the wheels. However since the rollator is already in motion, the inertia continues to push the rollator forward until the friction forces bring it to a stop. The same sort of delay can be seen in Figure 25 angular velocity plot around the 180th time step. With regards to velocity control, another issue is that the wheel velocities are bounded. Once the desired linear and angular velocities are computed, the forward kinematics is used to calculate the wheel velocities. When both the linear and angular velocity values are high, it is possible that the desired wheel velocity is higher than the upper bound of the wheel capability. In these instances, there can be a large error in the velocity response. The angular velocity response in Figure 26 is most probably due to this reason.

Future work for the control systems development includes incorporating the robot dynamics to achieve better velocity tracking. It is common practice to control wheel torque control using the dynamics to achieve more accurate velocity tracking. Another important aspect that needs to be addressed is to incorporate the human-robot interaction in the control design. There can often be a significant longitudinal/lateral input force from the user. For these instances, it is important to track these inputs and implement it within our control systems to achieve desired distance-orientation control.

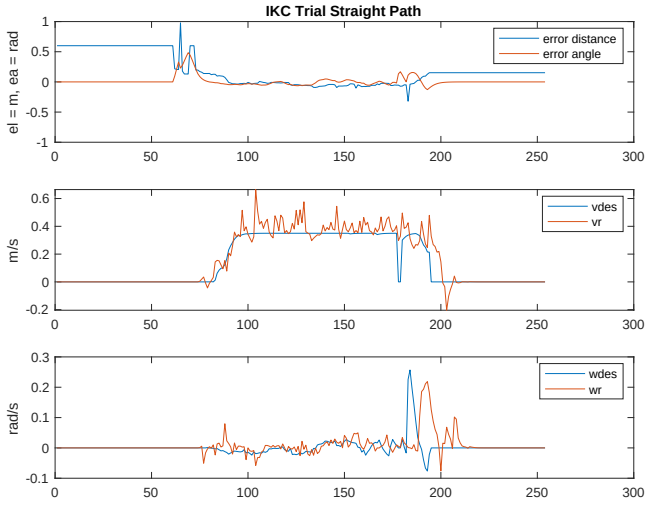


Figure 25: Experimental result: straight path

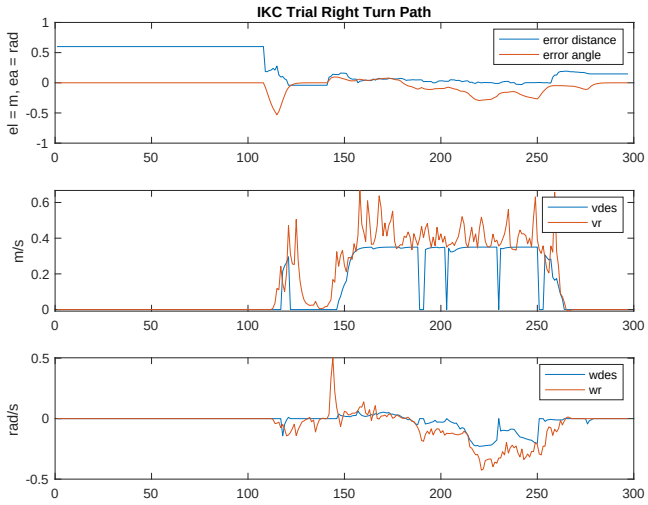


Figure 26: Experimental result: right turn

4 Human State Estimation

4.1 Human State Estimation

Estimation of human orientation using coaxial rgb-depth images. The objective of this chapter is to report the development of the human state estimation module to estimate the velocity, distance and orientation of the user as required by the control system, as discussed in Chapter 3.

4.1.1 Background

Various methods for human orientation estimation for assistive device control have been proposed, such as placing an IMU on the user [30, 31]. Requiring users to wear a sensor at all times is not practical for many users, requiring the user to remember to put the sensor on, which can be challenging for the elderly. Furthermore, sensor packages can be irritating to wear, leading to poor compliance and reliability. Furthermore, IMU drift associated with numerical integration can cause artifacts in estimation. While reset approaches have been suggested, it can be difficult for the system to reset as the rollator does not have sufficient knowledge of the human without using other sensors.

Vision-based approaches that utilize both RGB (colour) and RGB-D (colour + depth) have been proposed as alternative methods for human state estimation. Classical vision-based approaches use feature descriptors, such as Histogram of Oriented Gradients (HOG), and some form of classifier, such as decision trees, to make their estimations. Chen et. Al [32] used HOG along with a kernel formulation to estimate human body pose. Their best estimator had an average error of 19 degrees, but has performed as poorly as 37.7 degrees error. Shinmura et al. [33] treated orientation estimation as a classification problem where they used HOG features from RGB-D data and employed a multi-class support vector machines (SVM) to classify orientation into discrete bins. However, their bin size of 45 degrees is too coarse for the rollator user state estimation problem under consideration.

Recently, there has been substantial advancements in computer vision using deep learning. In particular, Deep Neural Networks (DNNs) have been shown promise to as extract feature given raw inputs compared to tailored feature extraction a priori. Choi et. al [34] viewed orientation estimation as a classification problem where they use a shallow Convolutional Neural Network (CNN) to extract features for a multi-class SVM classifier where orientation is classified into bins of 45 degrees. Once again, 45 degree bins are too coarse for the rollator control system under investigation.

Orientation estimation can be tackled using the concepts from pose estimation. For our controls requirements, torso orientation can be estimated using the locations of upper body landmarks. In particular, by knowing the shoulder locations, depth data drawn from RGB-D cameras can be used to inform orientation calculations.

One approach to pose estimation is a keypoint detection approach, using deep learning (e.g., CNN) as a feature extractor followed by regression to predict the coordinates of the body segments. One of the first high performing pose estimators, DeepPose uses a cascade of Pose Regressors to predict joint location of body parts [35]. The initial network consists of 7 layers (5 convolution and 2 Fully Connected) which receives a 220x220 image as an input and regresses to a vector of length 2k where k is the number of body parts. Each subsequent network is trained using the prediction from the above network to create a bounding box around the predicted coordinate, which forms the cropped image input to the next network. The purpose of these networks is to identify a higher resolution image to refine its prediction. Considering their network achieved a Percentage of Correct Parts accuracy of 56% for an upper body pose, regressing directly to the coordinates likely adds unnecessary complexity and weakens generalization [36]. Due to its low computational requirement, DeepPose is an attractive approach, especially considering the variability of expected poses is low (i.e., camera fixed to rollator frame). However, initial implementations of this algorithm for WATRR led to poor accuracy for our use.

An alternative approach has been using CNNs to generate body part heat maps instead of directly regressing to a coordinate [37]. The location of the joint is then found using a heuristic, such as the pixel with the highest probability. There are several benefits of predicting heat maps. The first and foremost advantage is improving the generalizability of the network and thus the accuracy [37]. Furthermore, the keypoint detection method 'forces' the network to output a coordinate, which means that even in the absence of a joint, one is predicted. Furthermore, it does not have sufficient degrees of freedom to predict multiple joints in the case of multiple people being in the frame.

4.1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) were popularized by LeCun et al. [38] in the 1980s/1990s, inspired by the human visual system where information from the eyes is relayed through neurons in a hierarchical manner [39]. Neurons are activated by certain features, with simpler neurons in earlier layers, whose activations are passed onto more complex neurons in a hierarchical manner. CNNs use this same concept to learn complex data through feature extraction, followed by combining features to make predictions.

CNNs layers can be categorized into 3 different categories: 1) an input layer, which are images in this chapter, 2) an output layer, which are the heat map predictions, and, 3) hidden layers between the input and output layer. Each of these carry out mathematical computations to learn features, which are passed onto the next layer and eventually an output.

Described below, the main layers of a CNN are convolutional layers, pooling layers and fully connected layers.

Convolutional Layers: The main building blocks of CNNs are convolutions which are computed using:

$$y[m,n] = x[m,n] * h[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[m,n] \cdot h[m-i, n-j] \tag{23}$$

where y is the convolution output, x is the input image and h represents the filters. The convolution operation here is represented by $*$. The filters, also called kernels, slides over the image at a given stride extracting features. The filter is essentially a matrix of learnable parameters. Typically at each convolution layer, there are multiple filters, with the number of filters being one of the design decisions for CNNs.

Typically convolution layers are followed by an activation function. The purpose of activation layers is to introduce non-linearity to the model. Convolution layers are linear operations, and the use of non-linear activation functions permits modeling of non-linear problems. Some of the commonly used activation functions are Tanh, Sigmoid, Rectified Linear Units (ReLU), and Softmax and its variations. For our estimator, ReLU was chosen as it is used in the models which our estimator is based on [40]. One of the main benefits of ReLU is that it has shown to converge more quickly than other functions.

Pooling Layers: Pooling layers are an integral component of CNNs, typically found between convolution layers. The purpose of pooling layers is to downsample the features for further processing and also extract dominant features. Average pooling and max pooling are most common pooling layers used for image based CNNS. In the pooling operation, a matrix block of defined size slides over the features at a defined stride. For the average pooling, the output is the average value of the feature values at the block whereas for max pooling, it is the maximum value.

Fully Connected Layers: Fully connected layers (FCL) are typically found at the end of fully convolution networks. All nodes in a FCL are connected to all nodes in the previous layer. The input to the layer is multiplied by a weight vector W and an optional

bias vector, B, may be added. FCLs are useful at the end of networks for combining the features and make a classification or regression. One of the drawbacks of the FCL is it is computationally heavy as all nodes are connected to each either.

Loss Functions: An important design decision for any network is the loss function. In CNNs, the loss is the difference between the predictions to the labels/true values. This value is a measure of how close to the true values our network is able to predict, with the goal of minimizing losses during training. There are several different loss functions that are currently used, with mean absolute error (L1) and mean squared error (L2) being the most common for regression problems. These loss functions are calculated as follows:

$$L1 = \frac{\sum_{i=1}^{n} |y_p(i) - y_t(i)|}{n} \tag{24}$$

$$L2 = \frac{\sum_{i=1}^{n} (y_p(i) - y_t(i))^2}{n} \tag{25}$$

where y_p is the predicted values, y_t is the true value and n is the number of samples.

4.1.3 Dataset

Our models were trained on a mixture of datasets. One is the MPII set [41], a state-of-the-art publicly available pose estimation dataset. The MPII dataset is a challenging dataset that contains images with different backgrounds performing 410 activities. Furthermore, the dataset contains frames with multiple people and occluded joints. With regards to the rollator project, we only consider operation when both shoulders are visible, hence we only use images where both the shoulders are visible. In total, 1000 images from the MPII dataset were used for training. The MPII dataset also provides the center coordinate and the scale of the person in frame. For training, we use that information to crop the image. While previous papers recommend the scale factor should be multiplied by 200 to find the bounding box dimension, this procedure often leads to parts of the body being cut out in our experience. Therefore, we use a bigger bounding box with the following specifications:

$$\begin{aligned}
\text{Box Side} &= \text{Scale x 200 x 1.25} \\
\text{x top left} &= \min(0, \text{x center} - \text{Box Side}/2) \\
\text{y top left} &= \min(0, \text{y center} - \text{Box Side}/2) \\
\text{x bottom right} &= \max(\text{image width}, \text{x center} + \text{Box Side}/2) \\
\text{y bottom right} &= \max(\text{image height}, \text{y center} + \text{Box Side}/2)
\end{aligned}$$

In addition to the pose estimation dataset, a publicly available face detection dataset called Helen [42] was used. While most of the images are close-ups of faces only, there are images with the shoulders also in view. The dataset was reviewed and a subset of 300 images with shoulders in view were labeled manually and added to the training set. These images were cropped about the center point with the largest square possible. Images were kept square to preserve the aspect ratio.

Another dataset used for training was the FLIC set [43]. This dataset contains 5003 images, taken from movies. Most of the images are of the characters upper body, often facing straight into the camera, which makes this set very useful for us as the rollator would have a similar viewpoint. The labels are used to crop the images such that the person is location at the center of the image.

Finally, an additional 1500 public domain images from various web search images were scraped and collected to add to the training set. The images were selected to be similar to expected views from the rollator viewpoint, looking directly at the person. The purpose of these images were to include training data with varying angles, as we expect to obtain from the rollator. These images were also manually labeled and cropped to the maximum possible size about the center point. Examples of the different datasets can be seen in Figure 27.

Figure 27: Dataset Examples

Data Augmentation: CNNs work best when a well balanced data set with a sufficient size is used to train the model. However, data collection can be a very expensive process

and limited data can affect the model performance drastically. A common method to expand the dataset is data augmentation. Data augmentation is a method to synthetically expand the dataset by applying transformations to the available dataset. For all training images, we perform 2 types of data augmentation: rotation and scaling of the images. Each image is randomly rotated in the range of $[\pm 5, \pm 40]$ or scaled in the range of [0.7,1.3] to augment the dataset.

One of the main benefits of data augmentation is reduction of overfitting, when a model learns the noise associated with the training data rather than learning a generalized model which also predicts new (unseen) data well. When using deep networks without a sufficient size of data, the network often suffers from overfitting.

Heat map Generation: The goal of the network is to predict an output heat map. The ground truth is a $64x64xn$ heat map where n is the number of body parts being predicted. To create the heat map, we create a 2D Gaussian centered around the labeled body part (x_c, y_c) with a standard deviation of 8 pixels. An example of the generated heat maps for a given image can be seen in Figure 28.

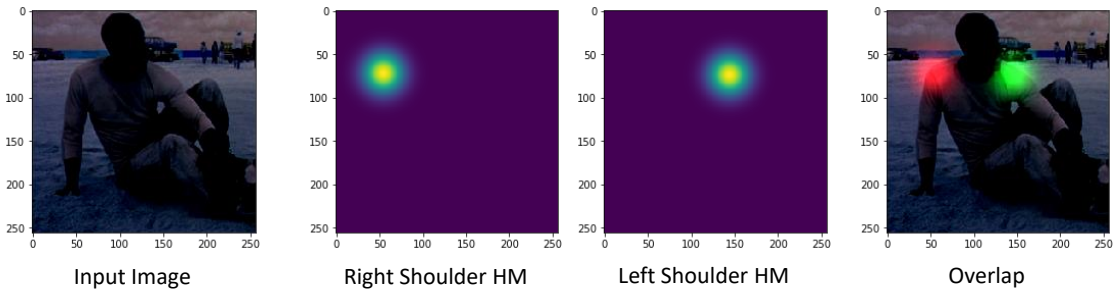


Figure 28: Sample heat map generation for the left and right shoulders using a sample input image

4.1.4 CNN Architecture

For our shoulder estimator, we modify the work of Newell Et. al [40] accordingly for our purpose. The input to the system is a 256x256x3 image and the output is a 64x64xn heat map where n is the number of body parts we would like to estimate, which for our case is 2 (i.e., right shoulder and left shoulder). The estimator predicts the shoulders independently, however the controller is designed such that if both shoulders are not in the viewpoint or

detected, the controller is set to brake state. The proposed network takes advantage of residual blocks, seen in Figure 29 which has been used by several methods recently [44,45].

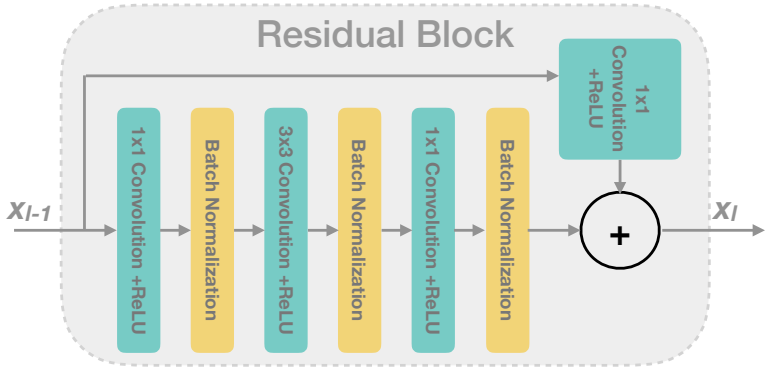


Figure 29: Residual module

The complete network architecture can be seen in Figure 30 in an 'Hourglass' structure. The network starts off with a front module, which consists off a $7x7$ convolution layer which a stride of $2x2$, followed by a residual block, a $2x2$ max pooling layer and then 2 more residual blocks. This downsamples the image from $256x256$ to $64x64$. The final output is $64x64$ resolution since upsampling all the way to $256x256$ can be computationally expensive, with only minor losses in accuracy.

After the front module, a series of residual blocks was applied followed by a max pooling layer to downsample to a minimum resolution of $4x4$. A series of residual blocks is applied with nearest neighbour upsampling after each block to increase resolution. In the downsample stream, the output of each residual block is branched to another residual block whose output is added to the output of each upsampling block. This structure has shown to be able to make accurate predictions across multiple scales. Once the desired resolution is achieved, an end module comprising of two $1x1$ convolution layers is applied, with the last convolution layers activation being linear. The output of the network is a partial heat map.

Training Details and Parameters: For training, the number of filters was set at each residual block to 192 for all layers. Adams optimizer is used with a learning rate of 1e-3, which is reduced by half upon plateauing, with a waiting period of 10 epochs. Our network was trained to 100 epochs.

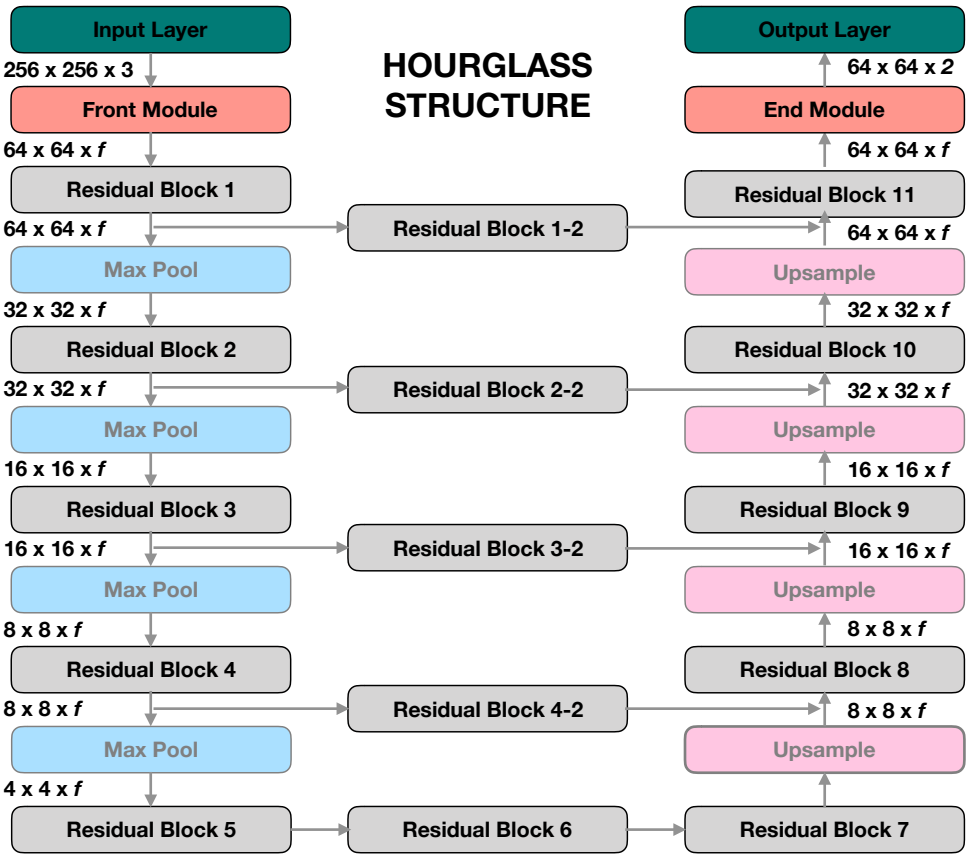


Figure 30: Hourglass structure

While L2 loss was initially applied for training, similar to [40], it was observed that the network kept converging to predict all outputs to 0. This is most likely due to the majority of the output map being black (given by a value of 0). Due to this, we propose a custom loss function, similar to the L2 loss, except false negatives are penalized by weighting them 100 times more than the background. L2 loss is given by:

$$L2 = \frac{\sum_{i=1}^{n}(y_p(i) - y_t(i))^2}{n} \tag{26}$$

whereas, the modified loss function is given by:

$$L = \frac{\sum_{i=1}^{n}(y_p(i) - y_t(i))^2 W(i)}{n} \tag{27}$$

where $W(i)$ is a weight matrix of size $64x64x2$, with the weights being 100 at the pixel locations where the generated Gaussian value is more than 0 and the weight being 1 at all other locations. This yields an asymmetrical weight biased to penalize false negatives.

4.1.5 State Estimation

The main purpose of this module is to estimate the human state (l, ψ, θ, v_h, w_h) as shown in Figure 10. These parameters are required for the IKC controller to generate the desired velocities based on the human-robot kinematics. The first step of the algorithm filters out the image background in order to improve prediction accuracy as well as ignore anyone visible in the background. This is done with the aid of the depth data generates 3D position data (i.e., X,Y and Z directions). This data is used to create a simple mask to mask out image pixels which are not in between $0.1 - 1.2m$ in the XY plane, as follows:

$$\begin{aligned} mask = 0.1m < depth < 1.2m \\ image = image. * mask \end{aligned} \tag{28}$$

That image is then fed into the CNN as described above. Low confidence predictions are filtered out, as well as any prediction locations that fall in the background.

Once we have the filtered outputs, we find both the x and y distance of the person's left and right shoulders respective to the world coordinates by indexing the locations with high confidence. The camera is set such that the camera is parallel to the ground, however a different orientation can also be used along with a rotation matrix to find the x and y distances respectively. A simple outlier rejection is conducted to find distances, d_f, by ignoring all data that outside one standard deviation.

Finally, the mean of d_f is used for calculating l, θ and ψ, shown in Figure 10. The distance from the camera to the person's center is calculated using $l = (d_{x,avg}^2 + d_{y,avg}^2)^{1/2}$ where $d_{x,avg} = \frac{d_{x,ls}+d_{x,rs}}{2}$ and $d_{y,avg} = \frac{d_{y,ls}+d_{y,rs}}{2}$. The angle between the rollator x-axis and the distance line is defined by θ and found using $\theta = \arctan(\frac{d_{y,avg}}{d_{x,avg}})$. Finally, to find ψ, the orientation of the person respective to the rollator is estimated by $\zeta = \arctan(\frac{\Delta d_x}{\Delta d_y})$ where

$\Delta d = d_{rs} - d_{ls}$. Further, ψ is found to be $\psi = -\theta + \zeta$. In the above equations ls and rs denote the left and right shoulder, respectively. Both ζ and θ are then filtered using a simple limiting filter followed by a low pass filter:

$$d_{filt,t+1} = Ad_{filt,t} + Bd_t \tag{29}$$

where A and B are constants that define the time delay and d is the data being filtered.

The human linear and angular velocity is then calculated using the following equations:

$$\dot{l}_{t+1} = \frac{l_{t+1} - l_t}{\Delta t} \tag{30a}$$

$$\dot{\psi}_{t+1} = \frac{\psi_{t+1} - \psi_t}{\Delta t} \tag{30b}$$

$$v_{h,t+1} = \frac{\dot{l}_{t+1} - v_{r,t+1}\cos\theta_{f,t+1} + kw_{r,t+1}\sin\theta_{f,t+1}}{-\cos\psi_{t+1}} \tag{30c}$$

$$\omega_{h,t+1} = \dot{\psi}_{t+1} + \frac{v_{r,t+1}\sin\theta_{f,t+1} + kw_{r,t+1}\cos\theta_{f,t+1} + v_{h,t+1}\sin\psi_{t+1}}{l} \tag{30d}$$

where Δt is the sampling time and k is the length of between the back wheels and the camera.

4.1.6 Experimental Results

To validate our CNN, a set of 200 images from the public domain were acquired. These images were selected as they closely resemble the camera view point from the rollator. For about half of these images, the background was subtracted completely, similar to expected ratios using the depth data as mentioned in the above section.

To evaluate the shoulder estimator, the location with the highest predicted value for the estimated shoulder position was found for each image. We then calculate the distance between the predicted and the actual pixel locations. If the distance is below a certain (error) threshold, we consider the estimation as being predicted correctly. The reason behind not using the exact labeled pixel is that most of the times, the shoulder is not located at a single point, but rather across various pixels. Furthermore, depending on the how close the person is to the camera, the shoulder size will vary. The results for different distances from our model can be can be seen in Table 3.

Table 3: CNN Accuracy for different distance thresholds

	10	**15**	**20**	**25**	**30**
Left Shoulder	62.5%	83%	90%	92%	93%
Right Shoulder	64.5%	82%	89.5%	95%	95.5%

Sample result images from the CNN is shown in Figure 31. The images indicated by the green box (i.e., top 2 rows) were counted as correct with a distance threshold of 30 pixels, whereas images in red box (i.e., bottom row) were considered incorrect. From the images, it can be observed that some of the locations that were deemed to be incorrect, may actually be considered correct when evaluated by visual inspection. For example, the second image from the left can be considered correct from visual inspection, however since the label was further than 30 pixels away, it was did not meet the accuracy criterion and labeled as incorrect.

Figure 31: Shoulder estimation heat map prediction

To further evaluate our state estimator, experimental trials were carried out for our shoulder estimation module. The experiments were carried out in a lab setting with a healthy 23 year old female. The lab was set up with Vicon motion capture cameras at

100Hz with simultaneous image-based estimator module at 8Hz. Four (4) markers were tracked: 2 on the camera and 2 on the person's shoulder (1 on each side). Two (2) types of trials were performed: 1) swinging shoulder back and forth (simulating walking) and 2) moving closer/further away from camera (to validate our length values).

Figures 32-33 show selected results from the trials. As seen from the figures, the distance predictions, l, for the trials were fairly accurate with an average RMSE of $16.53mm$ and $14.61mm$ respectively. As for ψ, the errors are $0.11rad$ and $0.13rad$.

Generally, both accuracy measures indicate strong evidence for use in estimating the human state. We observed from the graphs that there is a frequent amplitude loss. This can be due to several reasons including: 1) error associated with the camera depth data, and/or 2) error associated with Vicon motion tracking shoulder markers, 3) the error associated with averaging the area can also lead to discrepancies.

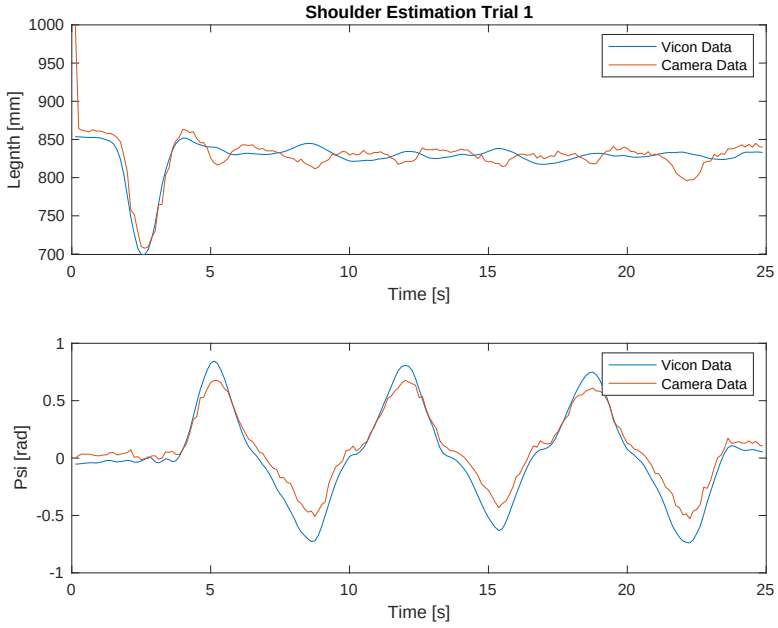


Figure 32: Shoulder estimation trial 1: l mRMSE = 16.53 mm, ψ mRMSE = 0.11 rad

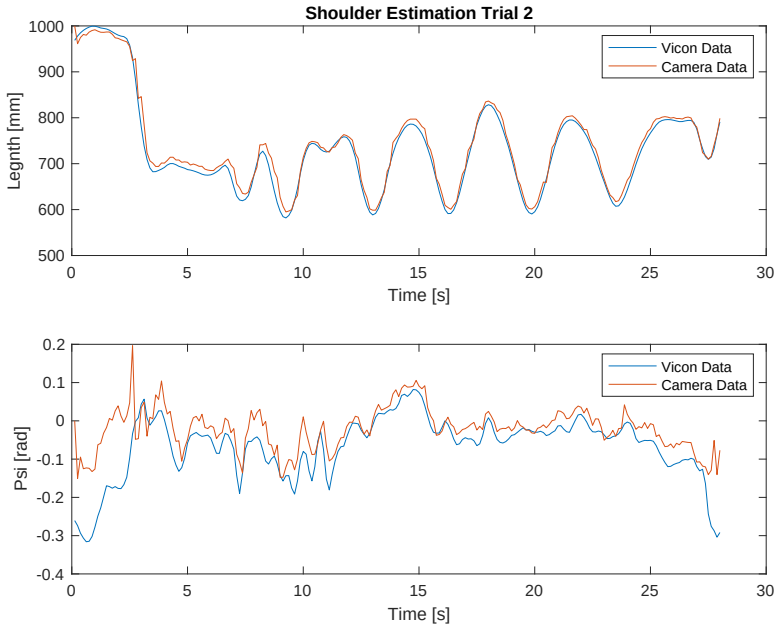


Figure 33: Shoulder estimation trial 2: l mRMSE = 14.61 mm, ψ mRMSE = 0.13 rad

5 Robot State Estimation

The following section has been submitted for journal publication and is currently under review.

Robot state estimation has also been identified as a key requirement for robotic rollator control. The states of interest are the position of the robot in world coordinates $P = \begin{bmatrix} x(t) & y(t) & \theta(t) \end{bmatrix}^\top$, the robot longitudinal and lateral velocities which are represented by v_x and v_y respectively in this section.

Lateral motion stability control is a key challenge in robotic rollators [18]. Typically elderly rollator users tend to have slower reaction times and lower strength leading to poor control of conventional rollators. To reduce these risks, advancing robotic rollators to smooth lateral trajectories and maintain desired orientation control have been proposed, motivating the need for accurate estimation of longitudinal and lateral velocity. While encoders and the robotic rollator (or 'robot' in this paper) kinematics can be utilized to estimate longitudinal velocity, lateral velocity estimation remains a challenge due to limitations in sensors and time-varying wheel force parameters. Recent studies for lateral velocity estimation of robotic rollators have focused on model-based approaches, which require resource-intensive instrumentation or difficult to estimate inertial and time-varying model parameters.

Zhang et al. used a model-based approach requiring difficult to measure tire forces and physical characteristics of the robot to estimate lateral velocity in a simulated environment only, with no experimental results reported [18]. Wada et al. designed a caster mechanism instrumented with encoders to estimate the steering angle used for the velocity estimation [46]. This approach requires additional mechanism and increases the mass of the system, which limits applicability for motor-impaired populations. While there exist related kinematic-based state (longitudinal and lateral speed) estimation approaches for mobile robots/vehicles [47–49], these approaches cannot be feasibly applied to light-weight rollators due to the expensive sensor components (i.e., high quality inertial measurement units (IMU) and force, torque, and steering measurement systems typically used in such applications). Moreover, such model-based methods are not challenged since mass and inertia parameters are not constantly changing due to time-varying vertical, longitudinal, and lateral loads (by the user), which play a key role in model-based observer models. These practical constraints of robotic rollators motivate the investigation of data-driven and learning-aided estimation methods.

Recently, there has been promising advancements in dynamical system identification and state estimation by deep learning [50], including applications for assistive technolo-

gies. Wang et al. proposed DeepSpeedometer, a method using a long short term memory (LSTM) based network to predict longitudinal speed using raw accelerometer and gyroscope data [51]. The method was able to outperform IMU-based kinematic approaches in various test cases with an average error of $< 0.5\%$. LSTM, residual network (ResNet), and Temporal Convolution (TCN) architecture backbones are used in [52] to estimate positions and orientation of individuals from IMU data. Although data-driven estimation approaches are promising methods for mobile robots and structured environments, their performance is considerably limited for robotic rollators due to changes in the loading conditions by the human/user and *corner cases* that require various excitation modes during the learning process. Hence, we propose a hybrid (learning-aided) estimation framework that augments an unscented Kalman filer (UKF) with deep learning components to address model uncertainties for speed estimation in this paper.

5.1 Preliminaries and State Observers

The rollator kinematics and model description, which are needed to develop a state estimator and enable design of a control system for assisting in longitudinal and cornering scenarios, are provided in this section. The WatRR test platform, the robotic rollator experimental setup in this paper (Fig. 34b), is modeled as a differential-drive mobile robot where the two rear wheels actuate and the front wheels are on casters (Fig. 34a).

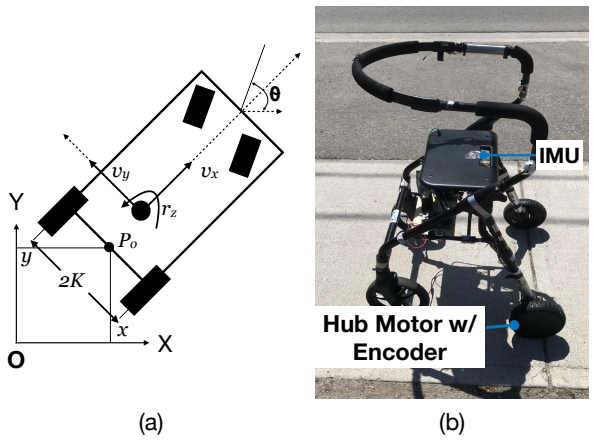


Figure 34: Experimental setup (a) rollator's kinematics and the coordinates (b) WatRR platform

5.1.1 Model Description

The forward kinematics of the rollator with the state vector $P = \begin{bmatrix} x(t) & y(t) & \theta(t) \end{bmatrix}^\top$ yields:

$$\dot{P}(t) = \frac{R_e}{2} \begin{bmatrix} \cos\theta & \cos\theta \\ \sin\theta & \sin\theta \\ K^{-1} & -K^{-1} \end{bmatrix} \begin{bmatrix} \dot{\psi}_r(t) \\ \dot{\psi}_l(t) \end{bmatrix}, \tag{31}$$

where the rollator heading angle is $\theta(t)$, and position components for the point P_o are $x(t)$ and $y(t)$. The wheel angular velocities and effective radius are denoted by $\dot{\psi}_i(t), i \in \{l, r\}$ and R_e, respectively, and K is half the width of the wheel base. Time derivatives of the lateral and longitudinal velocity in the body frame yields $\dot{v}_x = a_x + r_z v_y$ and $\dot{v}_y = a_y - r_z v_x$, where a_x, a_y, and r_z are the robot's longitudinal and lateral accelerations, and the yaw rate measured by IMU (in the local coordinates) at CG. The system model, based on the kinematics can be written as:

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + Bu(t) + \varrho^p, \\ y(t) &= Cx(t) + \varrho^m, \end{aligned} \tag{32}$$

where, $x = \begin{bmatrix} v_x(t) & v_y(t) \end{bmatrix}^\top$, $u = \begin{bmatrix} a_x(t) & a_y(t) \end{bmatrix}^\top$, the process/measurement uncertainties ϱ^p, ϱ^m are due to the measured acceleration and yaw rate $r_z^m, a_{x,y}^m$ that include quasi-constant (angular velocity and acceleration) biases as well as zero-mean Gaussian noises, and

$$A(t) = \begin{bmatrix} 0 & r_z(t) \\ -r_z(t) & 0 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{33}$$

The model is then discretized by step-invariant method, in which the discrete-time realization is approximated by:

$$A_k^d \triangleq \phi^A_{t_{k+1},t_k} \approx e^{A(t_k)T_s} \tag{34}$$

$$B_k^d \triangleq \int_{t_k}^{t_{k+1}} \phi^A_{t_{k+1},\tau} B(\tau) d\tau \approx \int_{t_k}^{t_{k+1}} e^{A(t_k)(t_{k+1}-\tau)} d\tau \tag{35}$$

$$C_k^d = C(t_k) \tag{36}$$

where $\phi^A_{t_i,t_j}$ is the continuous time state transition matrix, which can be expressed by the Peano-Baker series. The realization is assumed to not vary significantly in each interval $[t_k, t_{k+1}]$ (i.e., sampling time), which is valid for the robotic rollator and the learning-aided

framework with the sampling time $T_s = 0.004$s. As a result, the discrete-time system can be written as

$$\begin{aligned} x_{k+1} &= A_k^d x_k + B_k^d u_k + \varrho_k^p \\ y_k &= C_k^d x_k + \varrho_k^m, \end{aligned} \tag{37}$$

with process and measurement uncertainties ϱ_k^p, ϱ_k^m, which have uncorrelated covariances $Q_k = \mathbb{E}[\varrho_k^p, \varrho_k^{p\top}]$ and $R_k = \mathbb{E}[\varrho_m^p, \varrho_m^{p\top}]$, respectively.

5.1.2 Kinematic-based state observer

There are two main model-based observer design methods for robot/rollator lateral velocity estimation using state observers: dynamic- and kinematic-based approaches. The former uses a dynamical model requiring identification of various parameters including robot mass, inertia, and the cornering stiffness of the tires. Considering the mass, center of gravity and inertia of the rollator are heavily influenced by human user interaction effects, real-time identification of these parameters necessitates high-resolution force-measurement instrumentation, which increases the cost and maintenance time significantly. Moreover, saturation of tire forces due to the slip results in challenges linear observer methods and requires nonlinear observer design that is arduous due to changes in human vertical forces.

Alternatively, the kinematic-based methods [47, 49, 53–55], which do not require the robot inertial parameters, tire/wheel forces, and vertical load distribution, rely heavily on IMU data to deal with accumulated errors associated with numerical integration over accelerations. The main challenge includes noise in the input and measurement channels for low-cost IMUs, which are widely used. In this regard, a linear observer is designed in [53], where the estimated states are obtained by $\hat{x}_{k+1} = (A_k^d - K_k C_k^d)\hat{x}_k + B_k^d u_k + K_k r_{z,k}$ with the gain

$$K_k \triangleq \begin{bmatrix} 2a|r_{z,k}| \\ (a^2-1)r_{z,k} \end{bmatrix},$$

in which a is a tunable parameter to assign the convergence rate of the error dynamics. While this method is an improvement over pure integration methods, it is still sensitive to sensor noise in affordable IMUs for mobile robotic platforms [56].

Alternatively, a Kalman state observer is designed in this paper (as the model-based estimation component of the developed learning aided framework, L-ASE) and is augmented by a data-driven estimator. In this framework, a_x, a_y, r_z are measured by the

IMU and priori information on longitudinal speed (i.e., $\bar{v}_x$) is calculated by the encoder data and (discrete-time version of) a stable first-order filter $\dot{v}_x^- = \bar{A}v_x^- + \bar{B}s_x$, where $s_x \triangleq \frac{R_e}{2}(\dot{\psi}_l + \dot{\psi}_r)$ is from the forward kinematics Eqn (31). Using the discrete-time model description Eqn (37), the following observability matrix confirms that the model is observable (i.e., $\text{rank}(\mathcal{O}) = 2$) for non-zero yaw rates.

$$\mathcal{O} = \begin{bmatrix} 1 & 0 \\ 0 & r_z \end{bmatrix}. \tag{38}$$

Considering the state variable initial conditions cannot be reconstructed when the the observability matrix loses rank, the estimated states are reset when the yaw rates fall below a certain threshold r_z^{th}. Before designing a Kalman filter, the detectability of the state estimator and the boundedness of the error covariance is investigated in the following.

Proposition 1. *The system in Eqn* (37) *with known initial states is uniformly detectable.*

Proof: By definition, the pair $[A_k^d, C_k^d]$ in the linear time-varying discrete-time system (37) is uniformly detectable if there exists $0 \leq \gamma_1 \leq 1, \gamma_2 \in \mathbb{R}^+$ and $\exists \varepsilon, \zeta_2 \geq 0$, such that

$$\vartheta^\top \mathbb{V}(\zeta_2, \zeta_1)\vartheta \geq \zeta_2 \vartheta^\top \vartheta,$$

whenever $\bar{\phi}_{\zeta_1+\varepsilon,\zeta_1}\vartheta \geq \gamma_1\vartheta$ for some ϑ, ζ_1 [57]; $\mathbb{V}$ denotes the observability grammian and the following should hold for some γ_3 for this purpose:

$$\mathbb{V} = \sum_{k=\zeta_1}^{\zeta_2} \phi_{k,\zeta_1}^\top C_k^{d\top} C_k^d \phi_{k,\zeta_1}, \quad \mathbb{V}(\zeta_2, \zeta_1) \geq \gamma_3 I > 0, \tag{39}$$

whereas $\phi_{i,j} = \phi_{i,i-1}\phi_{i-1,j},\ \phi_{i+1,i} = A_i^d$ are state transition matrices for $i \geq j$. The system (37) with non-zero bounded yaw rate (due to the kinematic constraints) satisfies (39) for the observability grammian and is uniformly detectable.

This is a sufficient condition for implementation of an optimal variance (Kalman) filter, which will be integrated with a data-driven estimator and discussed in Section 5.3.

5.2 Data Driven Estimation

Recurrent and convolutional networks are first briefly introduced in this section, then the proposed data-driven architecture is described.

5.2.1 Recurrent and convolutional networks

While Recurrent Neural Networks (RNNs) have proven to be very useful for time series data by using input/output data from previous time steps to make its current prediction, RNN methods are challenging to train due to the well-known vanishing of gradient problems. This occurs in the back propagation process, where when the gradients become too small, thus, an update in the parameters have little effect on the output. To deal with the vanishing gradient issue in RNNs, Long short-term memory (LSTM) networks, a modified version of RNN which contains a memory cell in its hidden layer to remember past inputs have been proposed. An LSTM network is able to map an input of length $z = (z_1, ..., z_k)$ to an output of the same length $y = (y_1, ..., y_k)$ iteratively at each time step k [58]. The basic structure of the memory cell shown in Fig. **??**a consists of three gates layers: forget, input, and output gate layers.

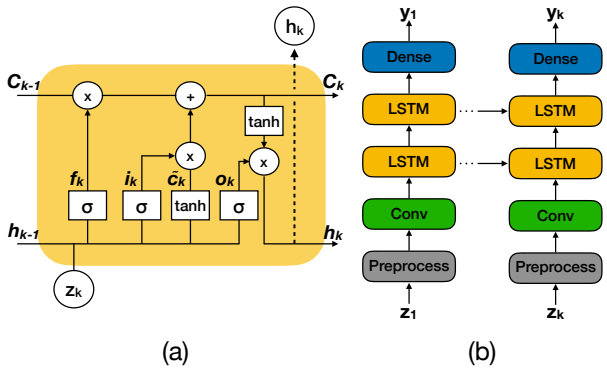


Figure 35: Learning Models; (a) LSTM memory block (b) Proposed network architecture

The forget gate layer contains a forget gate, f_k, which decides what information should be discarded from the cell state using $f_k = \sigma(W_{fh}h_{k-1} + W_{fz}z_k + b_f)$, where W_{fh}, W_{fz} are the weight vectors, h_{k-1} is the hidden state at the previous time step, z_k is an input vector, b is the bias vector, and σ is an activation function. The input gate layer decides what new information is to be added to the cell state, and its components are the input

gate state i_k and cell input state $\tilde{c_k}$, given by $i_k = \sigma(W_{ih}h_{k-1} + W_{iz}z_k + b_i)$ and $\tilde{c_k} = \tanh(W_{\tilde{c}h}h_{k-1} + W_{\tilde{c}z}z_k + b_{\tilde{c}})$, where tanh is an activation function. We can now calculate the cell output state c_k, as in (40), where $\odot$ denotes the element-wise product. The new cell output state combines information from the past with the input gate, and applies the forget gate to forget information as needed.

$$c_k = f_k \odot c_{k-1} + i_k \odot \tilde{c_k}. \tag{40}$$

The output gate layer includes the final hidden layer output (h_k) calculation and the output gate state o_k, which is calculated using the input/memory from the last time step:

$$o_k = \sigma(W_{oh}h_{k-1} + W_{oz}z_k + b_o), \tag{41a}$$

$$h_k = o_k \odot \tanh c_k. \tag{41b}$$

Temporal Convolutional Networks (TCN), on the other hand, have been practiced for various time dependent applications. The distinguishing feature of TCNs are their casual convolutions and ability to take a sequence of any length and map it to an output of the same length. In order to factor in the variable sequence length, TCN uses dilated convolutions which are able to have a larger receptive field than simple convolutions [59]. Larger dilation factors and larger filters can both be used to increase the receptive field. TCN uses residual blocks, which concatenate the transformations of input z to the initial input, in its network to avoid representational degradation [60]. A TCN model was developed for the robotic rollator states as it has shown note-worthy performance for sensor data regression problems. It was also used as a benchmark to examine whether fusing (augmenting) the data from a model-based approach (uncented Kalman observer) significantly improves performance. Since the tensors of the transformations and the input can be of different shapes, in our experiments a 1×1 convolution is performed on the input to ensure same tensor shape for the element-wise addition. In the developed TCN model, within the residual block, multiple dilated casual convolutions, each followed by a normalization, a ReLU, and a dropout layer were implemented (shown in the residual architecture in Fig. 36).

5.2.2 Network architecture

The proposed architecture for the C-LSTM (Convolutional-Long Short Term Memory) based learning algorithm can be seen in Figure 35b, where a stacked LSTM is depicted. Stacked LSTMs have the benefit of being able to build higher representations of sequenced data. The convolutional layers are added before the LSTM layers for feature extraction and it is shown in the results section that this improves the results.

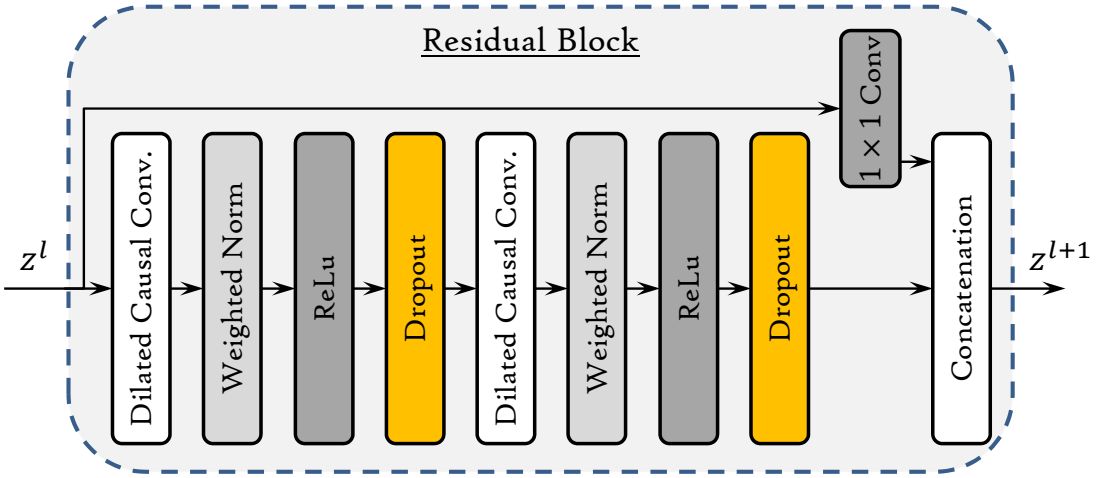


Figure 36: Residual block in the developed TCN model

The input vector $z_k \in \mathbb{R}^{N_{ts} \times 7}$ represents the measured IMU signals $a^m_{xy,k}, r^m_{z,k}$, wheel speeds $\dot{\psi}_{i,k}$, the difference between wheel speeds $\tilde{s}_x$, and priori longitudinal speed $\bar{v}_x$, as features, with a sliding window of $T_{ts} = 40$ms, which is obtained from several experiments ranging from 4 to 500ms. N_{ts} is the number of time steps taken into consideration. While it was observed that window length does not affect the estimator performance, increasing length drastically increased training time, justifying the choice for a small window (40ms). The input z_k of shape (samples, time steps, features) is fed into a convolutional layer, with a filter size of 256 and kernel size of 3, for feature extraction. The convolution layer maps it to z^l_k using $z^l_k = f_c(W_c * z_k + b)$, where W_c and b are learnable parameters, $*$ denotes convolution, and $f_c :=$ tanh is an activation function for our case. This is followed by two LSTM layers, each with 75 units and a dropout probability of 0.2, added to avoid over fitting. The output of this layer is then fed into a dense layer which maps the input h_t to an estimated state y_k employing $y_k = f_D(W_D h_k + b)$, in which W_D, b are the learnable parameters and $f_D := f_c$ is an activation function.

5.3 Proposed Augmented Method

To overcome challenges in the kinematic-based state observer design and the unobservablity issue during non-zero yaw rate, as well as corner cases in C-LSTM and TCN estimators, we propose L-ASE, a fusion framework shown in Fig. 37, in which the lateral speed prediction by learning is used as measurement in a UKF. The Gaussian noise characteristic, which is a required for the linear Kalman filter, cannot be assumed for such fusion mechanism due to utilizing data by the learning module in the measurement channel of the state observer. A better representation of such probability density function for non-Gaussian noise can be made using particle filters and UKF [61,62], which has been shown to be accurate at least to the second-order for non-Gaussian data [63].

In the unscented Kalman state observer, the state variable is approximated by Gaussian Random Variable (GRV), however specified using a carefully chosen set of sample points (sigma points) [63]. These points are said to capture the true mean and covariance of the GRV and when propagated through the system, it captures the posterior mean and covariance to the third order for Gaussian data and at least to the second order for non-Gaussian. The matrix $\mathcal{X} \in \mathbb{R}^{\mathcal{L}\times(\in\mathcal{L}+\infty)}$ captures the sigma points, where L is the dimension

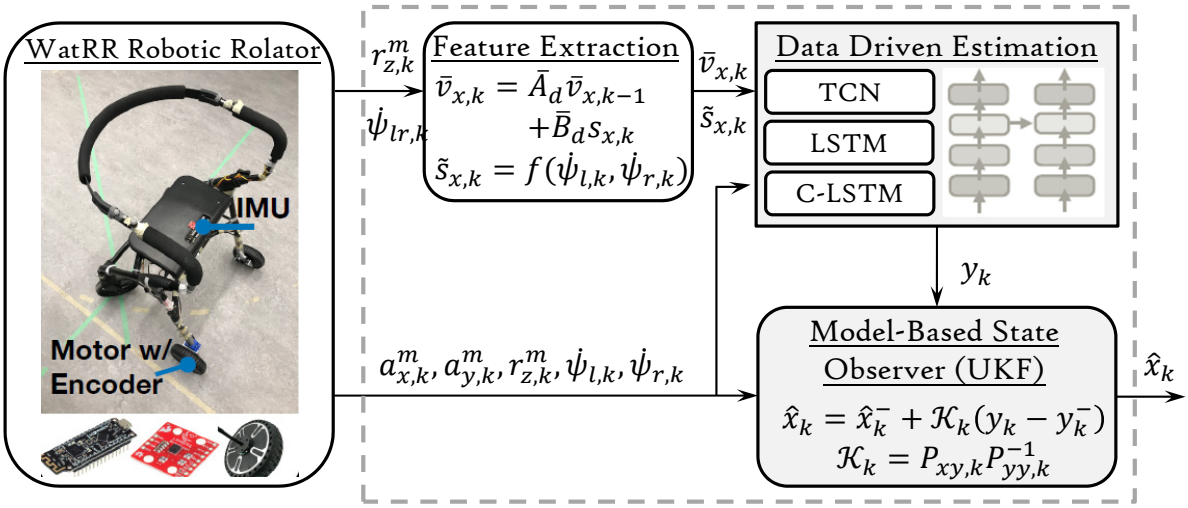


Figure 37: Learning-aided fusion architecture

of the state vector; it is initialized as $\mathcal{X}_{0,k-1} := \bar{x}_{k-1}$ and is given by

$$\mathcal{X}_{i,k-1} = \begin{cases} \bar{x}_{k-1} + [\sqrt{(L+\lambda)P_{k-1}}]_i & 1 \le i \le L \\ \bar{x}_{k-1} - [\sqrt{(L+\lambda)P_{k-1}}]_{i-L} & L+1 \le i \le 2L, \end{cases} \tag{42}$$

where $\lambda = \alpha^2(L + \kappa)$ is a scaling parameter, in which α determines the spread of the sigma point around $\bar{x}$ and κ is a secondary scaling parameter typically set to zero. In the prediction step, first the sigma points are propagated through $\mathcal{X}_{i,k|k-1} = f(\mathcal{X}_{i,k-1})$ and a priori state and covariance matrix estimate is made by

$$\begin{aligned} P_k^- &= \sum_{n=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \hat{x}_k^-][\mathcal{X}_{i,k|k-1} - \hat{x}_k^-]^\top + Q_k, \\ \hat{x}_k^- &= \sum_{n=0}^{2L} W_i^{(m)} \mathcal{X}_{i,k|k-1}, \end{aligned} \tag{43}$$

in which Q_k is the process noise and the weights are $W_0^{(m)} = \frac{\lambda}{L+\lambda}$, $W_0^{(c)} = \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta)$, and $W_i^{(c)} = W_i^{(m)} = \frac{1}{2(L+\lambda)}$; β is used to incorporate prior knowledge and is typically set to 2 for Gaussian distributions [63]. χ was then transformed through the measurement model, $\mathcal{Y}_{i,k|k-1} = h(\mathcal{X}_{i,k|k-1})$, followed by calculating the mean and covariance matrices $y_k^-, P_{yy,k}, P_{xy,k}$ described in Algorithm 1, time and measurement updates. The state and covariance estimates are updated as in

$$\hat{x}_k = \hat{x}_k^- + \mathcal{K}_k(y_k - y_k^-), \tag{44a}$$

$$P_k = P_k^- - \mathcal{K}_k P_{yy,k} \mathcal{K}_k^\top, \tag{44b}$$

where $\mathcal{K}_k$ is the iterative gain factor provided in Algorithm 1. UKF requires a small set of sample points, $2n + 1$, to approximate the mean and covariance of the output. Non-Gaussian input approximations can be improved to higher-order accuracy through the choice of α and β.

Remark 1. *The hypothetical advantage of the proposed augmented framework is addressing corner cases by utilizing the bounded error covariance property of the kinematic-based lateral speed (sideslip) estimator.This is guaranteed by the uniform detectability* $\mathbb{V}(\zeta_2, \zeta_1) \geq \gamma_3 I > 0$ *and observability of the system* (37)*, bounded/stable estimation error variance of the optimal stochastic filter* (44)*, and prioritizing process or measurement (i.e., data-driven estimates) through adapting* Q_k, R_k *covariances.*

Moreover, the advantage of UKF over KF for the developed robotic rollator speed estimator is investigated and discussed in the next section.

Algorithm 1: Learning-Aided Velocity Est. Algorithm

Input : $a^m_{x,y}, r^m_z, \dot{\psi}_l, \dot{\psi}_r$
Output: $\hat{v}_x, \hat{v}_y$
Initialize Kalman based Obs. with $\hat{x}_0 \triangleq \mathbb{E}[x]$, $P_0 \triangleq \mathbb{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^\top]$
while $t \geq 0$ **do**
i) $z_k \leftarrow$ pre-process$(a^m_{xy,k}, r^m_{z,k}, \dot{\psi}_{l,k}, \dot{\psi}_{r,k})$
ii) Long. speed priori $\bar{v}_{x,k} = \bar{A}_d \bar{v}_{x,k-1} + \bar{B}_d s_{x,k}$, by kinematics, $s_k = f_v(\dot{\psi}_{l,k}, \dot{\psi}_{r,k})$
C-LSTM trained network:
$z^l_k \leftarrow f_c(W_c * z_k + b)$ by CNN
$h_k = o_k \odot \tanh c_k$ by LSTM
$y_k \leftarrow f_D(W_D h_t + b)$ by Dense layer
Prediction and time update:
Sigma points $\mathcal{X}_{k-1}$ by (42)
$\mathcal{X}_{k|k-1} = f(\mathcal{X}_{k-1})$
Priori state/covariance estimates $\hat{x}^-_k, P^-_k$ by (43)
Transform $\mathcal{X}_{k|k-1}$ through $\mathcal{Y}_{k|k-1} = h(z_t, \mathcal{X}_{k|k-1})$
$y^-_k \approx \sum_{n=0}^{2L} W^{(m)}_i \mathcal{Y}_{i,k|k-1}$
Measurement update
$P_{yy,k} \approx \sum_{n=0}^{2L} W^{(c)}_i [\mathcal{Y}_{i,k|k-1} - \bar{y}_k][\mathcal{Y}_{i,k|k-1} - \bar{y}_k]^\top + R_k$
$P_{xy,k} \approx \sum_{n=0}^{2L} W^{(c)}_i [\mathcal{X}_{i,k|k-1} - \bar{x}^-_k][\mathcal{Y}_{i,k|k-1} - \bar{y}_k]^\top$
$\mathcal{K}_k = P_{xy,k} P^{-1}_{yy,k}$ and covariance Est. by (44b)
$\hat{x}_k = \hat{x}^-_k + \mathcal{K}_k(y_k - y^-_k)$
if $|r^m_z| < r^{th}_z$ **then**
$\hat{x}_k[1] = 0$;
return $[v_x\ v_y]^\top \leftarrow \hat{x}_k$
end

5.4 Experimental Evaluation

The proposed state estimation framework is experimentally verified in various maneuvers using the WatRR platform shown in Fig. 34. WatRR is a standard rollator equipped with two brushless DC hubmotors with built-in encoders for the rear wheels. The motor encoders have a resolution of 4.35 pulses per degree and provides wheel rotation data to calculate wheel speeds $\dot{\psi}_l$ and $\dot{\psi}_r$. An IMU with three-axis accelerometer, gyroscope and magnetometer, is used to collect longitudinal acceleration a_x, lateral acceleration a_y and yaw rate r_z. Data acquisition is performed with an Arduino Uno, which collects data at 250Hz through a serial interface. The visual verification and data acquisition setup with a Vicon motion capture system is shown in Fig. 38.

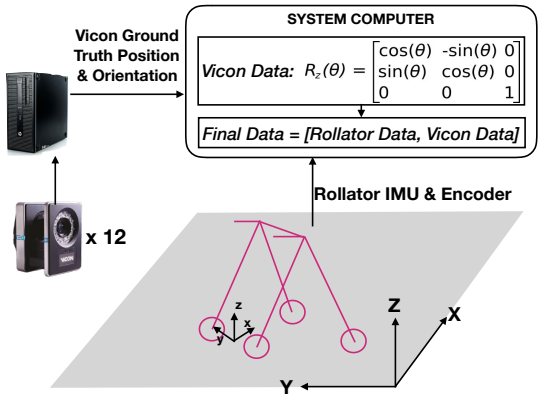


Figure 38: Visual verification setup and the test platform

5.4.1 Dataset details

Training data was collected at the University of Waterloo's RoboHub, set up with twelve Vicon motion capture cameras to collect the ground truth data. For data collection, three adults were asked to perform four different types of maneuvers (i.e., left turn, right turn, random cornering, and walk straight) with large speed variances and various artificially slowed/impaired gaits resembling limited mobility. Users were free to walk in their normal gait pattern, and were encouraged to walk at different walking speeds and turning radii across trials. In total, 27 sets of trials were performed and to increase the data set, the collected data was augmented, in which trials were scaled randomly with a factor of $[0.7, 1.3]$. The training set consisted of approximately 210k data points, with 80% of the

data was used for training and 20% for validation. For testing, 6 additional trials were used, which was not part of the training or the augmented datasets.

5.4.2 Performance evaluation and metrics

Based on the controller requirements to follow the human intent and minimize the human-robot relative distance (and heading), the functional requirements for the estimated lateral speeds necessitates a standard deviation bounded by $\sigma_{vy} < 0.015$ [m/s]. To evaluate the proposed learning-aided state estimation method, the mean root mean squared error (RMSE) and the average error percentage (AEP) criteria are used:

$$\text{RMSE} = \sqrt{\frac{\sum_{t=0}^{T}(\hat{y}_t - y_t)^2}{T}}, \tag{45a}$$

$$\text{AEP} = 1 - \frac{\sum_{t=0}^{T}|\hat{y}_t - y_t|}{\sum_{t=0}^{T}|y_t|}, \tag{45b}$$

The *Adam* optimizer with the mean squared error loss function, a learning rate = 0.0001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$, was used across all models which were trained to a maximum of 50 epochs and a batch size of 512; when the validation loss saturated but the training loss continued to decrease, early stopping was used to avoid overfitting.

The results of various explored data-driven and learning-aided (i.e., augmented) state observers explored are shown in Table 4. An estimator with a lower RMSE and a higher AEP is considered better performance over several tests. In a linear Kalman state observer (KF) on the model description (32), accelerometer noises cause a fairly large drift in estimation leading to poor performance, which was marginally improved by continuously resetting the state observer at low speeds ($v_x < 50$mm/s). Both data-driven and learning-aided models outperform the pure KF model significantly. The results also suggest that the CLSTM performs the best out of the data-driven models; the only difference between the CLSTM and LSTM method is the additional convolution layers. One plausible reason for the improvement in the CLTSM method over the LSTM is that the convolution layer is able to learn meaningful features over the pure LSTM method. Meaningful features are considered to be features/learned parameters which lead to a better mapping of the raw data/input to the output. While the network is a global optimizer and the exact filters are not known, filters can vary from smoothing to learning the bias of the signals. Furthermore, additional improvement is observed in the estimation with the learning-aided (i.e., L-ASE) approach. The augmentation of the data-driven estimator with the model-based KF observer (in the proposed framework) guarantees boundedness of the error covariance

Table 4: Estimator Performance Comparison

Method	**Controlled**		**Random**	
	RMSE (Std) [mm/s]	AEP	RMSE (Std) [mm/s]	AEP
KF	442.6 (420.0)	-	215.8 (188.3)	-
LSTM	12.9 (12.8)	0.692	19.0 (17.9)	0.759
LSTM w/ UKF	12.4 (12.3)	0.713	18.6 (7.7)	0.772
TCN	8.2 (7.9)	0.816	15.5 (15.0)	0.848
TCN w/ UKF	7.1 (6.3)	0.828	10.6 (10.2)	0.879
CLSTM	6.8 (6.6)	0.828	10.2 (9.8)	0.872
CLSTM w/ KF	6.8 (6.7)	0.827	10.1 (9.8)	0.873
CLSTM w/ UKF	6.8 (6.2)	0.832	10.0 (9.5)	0.877

and addresses edge cases accordingly; bounded and lower error covariance, especially during cornering with larger lateral velocities, constitutes better AEP. Such augmentation not only helps filter out noisy predictions in model-based observer design, but also addresses corner cases, which cannot be covered during training with different users and variations in vertical loads. This is due to the fact that LSTM, CLSTM and TCN do not implement model descriptions, leading to unreliable estimates in corner/edge cases. This limitation is addressed by utilizing the model knowledge as in KF/UKF in the proposed approach, as experimentally verified in the next subsection in Fig. 44.

5.4.3 Augmentation Performance and Ablation Study

Examples of a straight, left-turn, and right-turn trials using the augmented CLSTM method are shown in Figs. 39-41. The speed estimation results based on the wheel encoder are denoted by S-WR, and as can be seen from experiments, it results in non-smooth as it amplifies the measurement noise associated with the encoder data.

An observation was that the acceleration data followed the same trend as the velocity estimates. For example, from 0-2 seconds in Figure 39, the longitudinal acceleration increased from zero and is positive which corresponds with the user speeding up. The longitudinal acceleration follows a cyclic trend around the $0\frac{m}{s^2}$ mark, corresponding with the users step frequency. Lateral acceleration excitations at the beginning due to the rollator wheels correcting on initial movement were observed, and also around the 2.5s mark

when the user starts turning. A similar trend can be observed from the right turn trial in Fig. 40 where the longitudinal acceleration and velocity follows a periodic pattern in correspondence to the user's gait frequency.

In the straight maneuver trial in Fig 41, the user started moving around the 0.5s mark, indicated by lateral and longitudinal acceleration and velocity excitations. The user then slowed down around the 2.5s mark, indicated by a negative longitudinal acceleration before speeding back up around 4.75s. To further validate the robustness of the estimator, users were also asked to perform random maneuvers including straight and harsh cornering scenarios, as discussed in the following.

In Fig 42, the user walked straight from 0.5s-4s before stopping and resting for about 2 seconds. The user also pulled the rollator slightly around the 5s mark resulting in minor (negative) longitudinal speed in the body frame. Moreover, the user turned slightly right and then left around the 7s mark, observed in both the lateral acceleration and velocity profiles. In another random-walk experiment, the user took a harsh left turn around 2s, as depicted in Fig. 43. The lateral velocity for this case reached as high as 0.18 [m/s]. Despite the harsh turn, the developed L-ASE estimator is still able to predict the velocities in the body frame accurately. While the velocity and acceleration data followed a similar trend, relying solely on acceleration data to estimate the state leads to poor performance due to sensor noise. It can be observed that the developed learning-aided models are able to estimate the lateral states reliably even with the presence of acceleration noises.

The benefit of the augmented approach is much more evident when compared to LSTM and TCN methods, as shown in Fig. 44. The proposed method guarantees stable estimation error covariance (by uniform detectability) and results in smoother estimates by propagating the sigma points in unscented transformation throughout the discrete-time process.

In addition to rejecting measurement noise, the system dynamics in the proposed approach deals with the corner cases more accurately than the pure TCN and LSTM, attributable to the lack of corner case training data. The predicted lateral velocity by the LSTM method is fairly noisy and has multiple spikes (around the 5.5s and 6s points), as shown in Fig. 44, due to inefficiency of the dataset in various rollator excitation conditions; this is filtered out by the L-ASE method by incorporating the system dynamics. Similar trends can be seen in the TCN method around the 5.5s mark.

The effect on the performance due to the network depth, manually selecting features by expert knowledge, the size of the sliding window and the number of LSTM units are studied in this subsection. Table 5 summarizes the results for a one- up to four-layer LSTM models, where all were trained with the same inputs for 50 epochs with a learning rate of 0.0001. As seen from the table, the two and three layer networks outperformed the

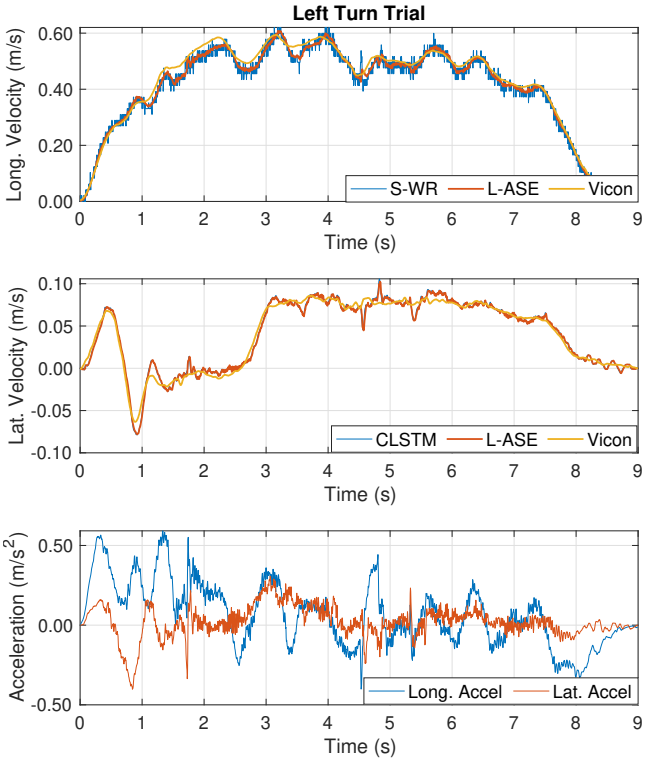


Figure 39: State estimation performance in a left-turn scenario

other networks. A network depth of 2 layer was selected as it has a lower computational requirement while similar results as the 3 layer network.

Stacked LSTMs enables a more complex hierarchical representation of our data. In our case, the stacked LSTM is able to extract temporal relations in the sensor data over a larger timescale. Since the velocity at time step t is affected by the states in the past, more complex temporal relations can improve the performance of the model. However, too deep of a network may use too large of a timescale to afford accurate estimation [51].

Another important observation to be made is that manually selecting features can improve performance. While convolutional networks are typically able to extract features automatically, it is often unable to extract simple features, such as the difference between wheel speeds $\tilde{s}_x$, and longitudinal velocity. By using the system dynamics knowledge and including such features, better velocity estimate can be made.

Furthermore, it is important to note that explicitly adding temporal features can actually harm the network performance. Both TCN and LSTM networks are expected to

Table 5: LSTM Depth and Features Effect

Method	**Controlled**		**Random**	
	RMSE [mm/s]	AEP	RMSE [mm/s]	AEP
1 Layer LSTM	8.1	0.794	13.3	0.837
2 Layer LSTM	6.8	0.822	9.8	0.879
3 Layer LSTM	7.0	0.817	9.5	0.886
4 Layer LSTM	8.2	0.786	10.5	0.866
Raw Features	7.7	0.812	10.6	0.870
RF & dT	9.9	0.756	11.4	0.862
RF & $(\mathrm{v}_x, \tilde{s}_x)$	6.8	0.822	9.8	0.879
8ms	6.9	0.822	11.4	0.870
20ms	7.0	0.820	11.6	0.865
40ms	6.8	0.822	11.4	0.865
80ms	6.9	0.818	11.2	0.865
160ms	6.7	0.830	11.1	0.873
10 LSTM units	7.2	0.820	11.5	0.855
25 LSTM units	8.3	0.798	12.2	0.851
50 LSTM units	6.8	0.829	9.6	0.884
75 LSTM units	6.8	0.822	9.8	0.879

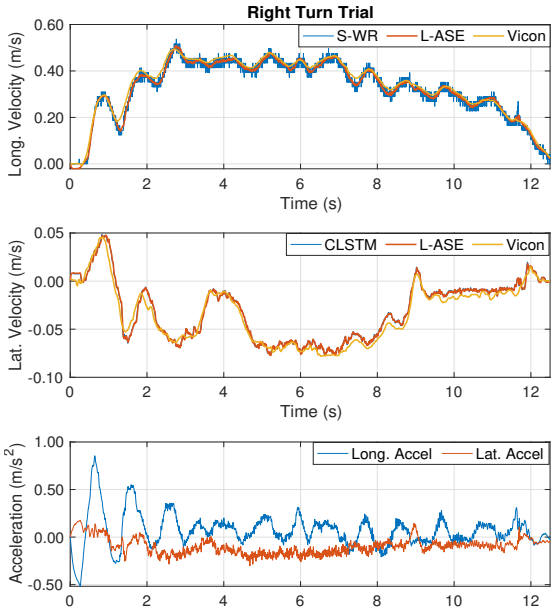


Figure 40: States and acceleration, right-turn assisted walking

capture the temporal relationships on its own. By adding the temporal data, we are essentially adding meaningless features which is harming the performance. Lastly, the effects of the number of LSTM units are provided in Table 5, and an improvement in performance is observed until 50 units, after which the performance difference is not notable.

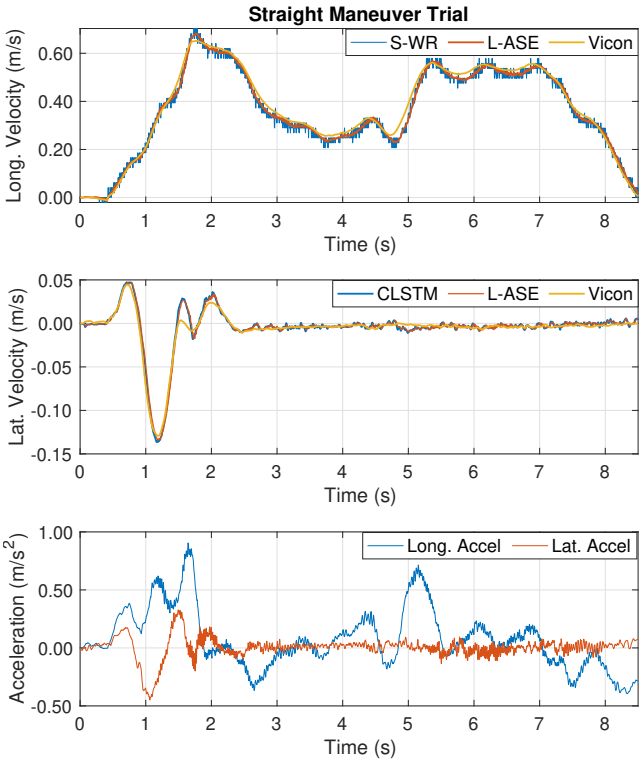


Figure 41: States and acceleration for straight/cornering scenarios

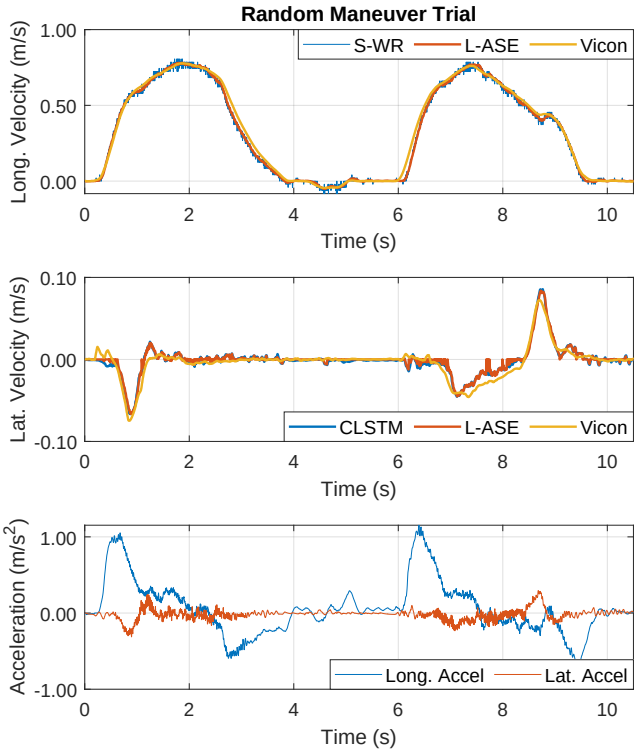


Figure 42: Learning-aided estimator performance in a random maneuver including cornering and straight assisted walk

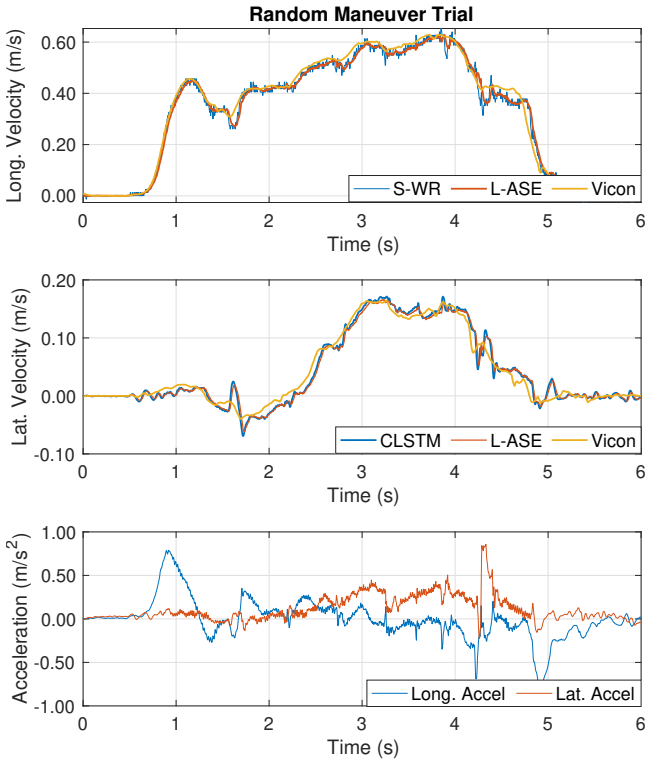


Figure 43: Learning-aided state estimation, random walk

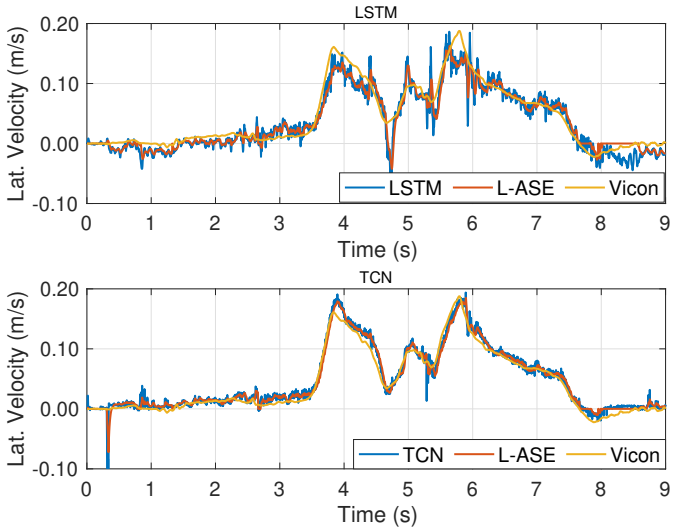


Figure 44: L-ASE performance comparison for Long./Lat. states

6 Conclusion and Future Work

As the aging population continues to grow, the demand for assistive technologies has risen along with it. Advancing assistive technologies, such as rollators, have been a focus of interest to aid individuals with impaired mobility. As the most popular device for mobility, rollators have been shown to improve quality of life through promoting independence and physical activity among its users. However, thousands of accidents and falls related to rollator use occur each year [16], most commonly due to improper use or rollator slipping away. There is also a large number of individuals who abandon their device soon after receiving it due to the fear of falls and difficulty with safe use.

To improve upon existing rollators and encourage its use, we propose a robotic rollator approach to implement control systems to assist and improve the user experience. We propose an inverse kinematic control (IKC) approach to control distance and orientation between the user and device. To address shortcomings of the IKC controller, the existing controller was modified. To begin, a variable gain schedule was used to account for the controller's sensitivity to shoulder swing and irregular gait pattern. We also implement a velocity level PID controller to reduce the effects of unknown disturbances and modelling errors such as the use of incorrect parameters or wheel alignment. Finally, a position level PID controller was implemented for braking during emergency states.

To achieve the proposed controller, a human state estimator is required. A CNN approach to predict shoulder locations was developed to inform a human state estimator, combined with depth camera data. The CNN is based on a stacked hourglass structure with residual blocks, which captures features across multiple scales by downsampling to a small resolution (4 x 4) followed by upsampling back to 64 x 64 resolution. After each up sampling layer, the tensor is added to its corresponding downsampling layer. The final output is a heatmap of 64 x 64 x n, where n is the number of body parts we would like to predict, which is 2 (left and right shoulder) in the current implementation. Based on accuracy, a heat map approach is adopted to locate the shoulders in lieu of regressing to a coordinate directly.

Finally, a robot state estimator based on a learning-aided augmented kalman filter was designed and tested. To further improve our controller, it is important to ensure the lateral stability of the robot. Current approaches for lateral velocity estimation requires expensive and bulky instrumentation along with in depth knowledge of the system dynamics, which is difficult to estimate due to the dynamics of human-robot interaction of our system. Recent developments in machine learning are leveraged to estimate the lateral velocity based on noisy IMU data with excellent results (mRMSE: 8.1 mm/s) which a great improvement over a pure kalman filter (mRMSE: 338 mm/s).

Limitations of the current work include minor oscillations of rollator due to shoulders swinging. While the current controller is an improvement over previous work, there is still a need for further advancement. Furthermore the velocity control of the current controller has some error associated with it which needs to be improved. Another limitation of the current work is the lack of experimental data with older adults and users with impaired mobility.

Moving forward, there remains plenty of work to be done to improve robot performance. To begin, incorporating a smart handle for generation of smoother trajectory is crucial to have a system with better user experience. Due to the complex nature of the human-robot interaction, it can be difficult to generate a trajectory which follows the intended path of the user, and is an area of research that needs to be explored in depth. In addition, the robot lateral velocity estimator also needs to be implemented to the robot along with the existing control systems to improve safety performance.

We also plan to explore visual-inertial navigation for a more reliable state estimator, as well as global navigation and mapping purposes. Furthermore, it is important to continue to extend the scope of the hybrid controller to include obstacle avoidance. It is also desired to explore incorporating a cheap and compact method to add the system dynamics to the controller architecture to improve velocity tracking. Finally, the future work also includes testing our system with the aging population and acquiring feedback for further improvements.

References

[1] Em-316a brushless dc-motor driver 12-35v 10a.

[2] Depth camera d435, Mar 2021.

[3] D.H Metz. Mobility of older people and their quality of life. *Transport Policy*, 7(2):149 – 152, 2000.

[4] Jason Shafrin, Jeffrey Sullivan, Dana Goldman, and Thomas Gill. The association between observed mobility and quality of life in the near elderly. *PLOS ONE*, 12:e0182920, 08 2017.

[5] Jean-Pierre Michel. *WHO world report on Ageing 2015*. 11 2015.

[6] Rahaf Alsnih and David Hensher. The mobility and accessibility expectations of seniors in an aging population. *Transportation Research Part A: Policy and Practice*, 37:903–916, 02 2003.

[7] Catherine Okoro, Natasha Hollis, Alissa Cyrus, and Shannon Griffin-Blake. Prevalence of disabilities and health care access by disability status and type among adults - united states, 2016. *MMWR. Morbidity and mortality weekly report*, 67:882–887, 08 2018.

[8] Diane L Damiano. Activity, Activity, Activity: Rethinking Our Physical Therapy Approach to Cerebral Palsy. *Physical Therapy*, 86(11):1534–1540, 11 2006.

[9] Keith Syrett. Aging, access, and the costs of health care: Should canada query the qaly? *International Journal of Canadian Studies*, 47:41–56, 01 2013.

[10] Hamid Bateni and Brian E. Maki. Assistive devices for balance and mobility: Benefits, demands, and adverse consequences. *Archives of Physical Medicine and Rehabilitation*, 86(1):134 – 145, 2005.

[11] L. Miles. Physical activity and health. *Nutrition Bulletin*, 32(4):314–363, 2007.

[12] Geunho Lee, Takanori Ohnuma, Nak Chong, and Soon-Geul Lee. Walking intent-based movement control for jaist active robotic walker. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44:665–672, 2014.

[13] Samer Mohammed, Juan Moreno, Kyoungchul Kong, and Yacine Amirat. *Intelligent Assistive Robots*. Springer Internnational PU, 2015.

[14] Ulrich Lindemann, Michael Schwenk, Jochen Klenk, Max Kessler, Michael Weyrich, Franziska Kurz, and Clemens Becker. Problems of older persons using a wheeled walker. *Aging Clinical and Experimental Research*, 28(2):215–220, Apr 2016.

[15] Judy Stevens, Karen Thomas, Leesia Teh, and Arlene Greenspan. Unintentional fall injuries associated with walkers and canes in older adults treated in u.s. emergency departments. *Journal of American Geriatrics Society*, 57:1464–1469, 2009.

[16] K Riel, K Hartholt, Martien Panneman, P Patka, Ed Beeck, and Tischa van der Cammen. Four-wheeled walker related injuries in older adults in the netherlands. *Injury prevention : journal of the International Society for Child and Adolescent Injury Prevention*, 20, 04 2013.

[17] J. Paulo, L. Garrote, C. Premebida, A. Asvadi, D. Almeida, A. Lopes, and P. Peixoto. An innovative robotic walker for mobility assistance and lower limbs rehabilitation. In *2017 IEEE 5th Portuguese Meeting on Bioengineering (ENBENG)*, pages 1–4, 2017.

[18] Xin Zhang, Jiehao Li, Zhenhuan Hu, Wen qi, Longbin Zhang, Yingbai Hu, Hang Su, Giancarlo Ferrigno, and Elena De Momi. Novel design and lateral stability tracking control of a four-wheeled rollator. *Applied Sciences*, 9:2327, 06 2019.

[19] A. Morris, R. Donamukkala, A. Kapuria, A. Steinfeld, J. T. Matthews, J. Dunbar-Jacob, and S. Thrun. A robotic walker that provides guidance. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 1, pages 25–30 vol.1, Sep. 2003.

[20] J. Paulo, L. Garrote, A. Asvadi, C. Premebida, and P. Peixoto. Short-range gait pattern analysis for potential applications on assistive robotics. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 107–112, Aug 2017.

[21] Eleonora , Sibylle Thies, Laurence Kenney, David Howard, Ulrich Lindemann, Jochen Klenk, and Rose Baker. Objective measures of rollator user stability and device loading during different walking scenarios. *PLOS ONE*, 14:e0210960, 01 2019.

[22] Avital Fast, Fikre S. Wang, Ronald S. Adrezin, Marc A. Cordaro, Juan Ramis, and Julian Sosner. The instrumented walker: Usage patterns and forces. *Archives of Physical Medicine and Rehabilitation*, 76(5):484 – 491, 1995.

[23] Amey Pawar, Ashwin Shenoy, Samson Wartika, Srikant Thavre, and Kamlesh Sasane. Design and development of electric scooter with regenerative braking. 11 2019.

[24] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. volume 3, 01 2009.

[25] Birgit Graf. An adaptive guidance system for robotic walking aids. *CIT*, 17:109–120, 01 2009.

[26] Carlos Cifuentes G. and Anselmo Frizera. *Human-Robot Interaction Strategies for Walker-Assisted Locomotion (Springer Tracts in Advanced Robotics 115)*, volume 115. 03 2016.

[27] Zengin, Halit. Control oriented system modelling and instrumentation of intelligent walker-human systems, 2015.

[28] P. M. Meshram and R. G. Kanojiya. Tuning of pid controller using ziegler-nichols method for speed control of dc motor. In *IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012)*, pages 117–122, 2012.

[29] Muhammad Asif, Junaid Khan, Muhammad Safwan, and Muhammad Rehan. Feedforward and feedback kinematics controller for wheeled mobile robot trajectory tracking. *Journal of Automation and Control*, 3, 06 2015.

[30] H. Ahmed and M. Tahir. Improving the accuracy of human body orientation estimation with wearable imu sensors. *IEEE Transactions on Instrumentation and Measurement*, 66(3):535–542, 2017.

[31] S. Zihajehzadeh, D. Loh, M. Lee, R. Hoskinson, and E. J. Park. A cascaded two-step kalman filter for estimation of human body segment orientation using mems-imu. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6270–6273, 2014.

[32] C. Chen and J. Odobez. We are not contortionists: Coupled adaptive learning for head and body orientation estimation in surveillance video. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1544–1551, 2012.

[33] Fumito Shinmura, Daisuke Deguchi, Ichiro Ide, H. Murase, and Hironobu Fujiyoshi. Estimation of human orientation using coaxial rgb-depth images. *VISAPP 2015 - 10th International Conference on Computer Vision Theory and Applications; VISIGRAPP, Proceedings*, 2:113–120, 01 2015.

[34] Jinyoung Choi, Beom-Jin Lee, and Byoung-Tak Zhang. Human body orientation estimation using convolutional neural network, 2016.

[35] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014.

[36] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christopher Bregler. Efficient object localization using convolutional networks, 2014.

[37] V. Belagiannis and A. Zisserman. Recurrent human pose estimation. In *2017 12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)*, pages 468–475, 2017.

[38] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[39] Grace W. Lindsay. Convolutional neural networks as a model of the visual system: Past, present, and future. 2020.

[40] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation, 2016.

[41] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[42] Vuong Le, Jonathan Brandt, Zhe Lin, and Lubomir Bourdev. Interactive facial feature localization. 10 2012.

[43] Benjamin Sapp and Ben Taskar. Modec: Multimodal decomposable models for human pose estimation. In *In Proc. CVPR*, 2013.

[44] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2016.

[45] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution, 2018.

[46] M. Wada, K. Ichiryu, T. Iguchi, and R. Yoshida. Design and control of an active-caster electric walker with a walk sensing system (smart walker). In *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 258–263, July 2016.

[47] C. B. Low and D. Wang. Integrated estimation for wheeled mobile robot posture, velocities, and wheel skidding perturbations. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2355–2360, 2007.

[48] J. Yi, H. Wang, J. Zhang, D. Song, S. Jayasuriya, and J. Liu. Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation. *IEEE Transactions on Robotics*, 25(5):1087–1097, 2009.

[49] K. Berntorp. Joint wheel-slip and vehicle-motion estimation based on inertial, gps, and wheel-speed sensors. *IEEE Transactions on Control Systems Technology*, 24(3):1020–1027, 2016.

[50] M. Brossard, A. Barrau, and S. Bonnabel. Rins-w: Robust inertial navigation system on wheels. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2068–2075, 2019.

[51] Q. Wang, Y. Gu, J. Liu, and S. Kamijo. Deepspeedometer: Vehicle speed estimation from accelerometer and gyroscope using lstm model. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, Oct 2017.

[52] Hang Yan, Sachini Herath, and Yasutaka Furukawa. Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods, 2019.

[53] J. Farrelly and P. Wellstead. Estimation of vehicle lateral velocity. In *Proceeding of the 1996 IEEE International Conference on Control Applications IEEE International Conference on Control Applications held together with IEEE International Symposium on Intelligent Contro*, pages 552–557, Sep. 1996.

[54] J. Pliego-Jiménez, R. Martínez-Clark, and C. Cruz-Hernández. Orientation and velocity observers for unicycle mobile robots. In *2019 18th European Control Conference (ECC)*, pages 2134–2139, 2019.

[55] L. De Pascali, F. Biral, M. Cocetti, L. Zaccarian, and S. Tarbouriech. A kinematic observer with adaptive dead-zone for vehicles lateral velocity estimation. In *2018 IEEE 15th International Workshop on Advanced Motion Control (AMC)*, pages 511–516, March 2018.

[56] Y. Liao and F. Borrelli. An adaptive approach to real-time estimation of vehicle sideslip, road bank angles, and sensor bias. *IEEE Transactions on Vehicular Technology*, 68(8):7443–7454, Aug 2019.

[57] BDO Anderson and John B Moore. Detectability and stabilizability of time-varying discrete-time linear systems. *SIAM Journal on Control and Optimization*, 19(1):20–32, 1981.

[58] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*, 2014.

[59] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *ArXiv*, abs/1803.01271, 2018.

[60] Y. Yang, J. Zhong, W. Li, T. A. Gulliver, and S. Li. Semi-supervised multi-label deep learning based non-intrusive load monitoring in smart grids. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2019.

[61] Badong Chen, Lujuan Dang, Yuantao Gu, Nanning Zheng, and Jose C. Prıncipe. Minimum error entropy kalman filter, 2019.

[62] Xujun Han and Xin Li. An evaluation of the nonlinear/non-gaussian filters for the sequential data assimilation. *Remote Sensing of Environment*, 112:1434–1449, 04 2008.

[63] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158, 2000.