"Wheelchair identification for Automatic docking solution for a wheelchair power assist"

Undergraduate Research Project

Aniket Khan
ME21B021

May 21, 2024

Contents

1

Problem Identification

1.1 Introduction

1.1.1 Powered Wheelchairs

Motorized wheelchairs offer significant benefits to individuals with mobility impairments, enhancing their quality of life and promoting independence. These wheelchairs provide users with greater freedom of movement, allowing them to navigate both indoor and outdoor environments with ease. With motorized propulsion, users can travel longer distances without exerting themselves physically, reducing fatigue and conserving energy. This increased mobility promotes social participation and engagement, enabling users to attend events, visit friends and family, and access community resources more easily. Additionally, motorized wheelchairs often come equipped with advanced features such as customizable seating, adjustable speeds, and obstacle avoidance systems, further enhancing user comfort and safety. Overall, motorized wheelchairs play a vital role in improving the mobility, autonomy, and overall well-being of individuals with mobility challenges.

Powered wheelchairs come in various types, each designed to meet specific mobility needs. Some of these types include:

- **Standard power wheelchairs:** Equipped with a joystick or control panel for basic mobility functions.
- **Mid-wheel drive wheelchairs:** Offer enhanced maneuverability and stability, ideal for navigating tight spaces.
- **Heavy-duty power wheelchairs:** Designed for users requiring higher weight capacities and durability, suitable for outdoor use.
- **Standing power wheelchairs:** Allow users to transition from seated to standing positions, promoting better circulation and posture.
- **Specialty power wheelchairs:** Customized to accommodate specific medical conditions or functional limitations, with features like reclining seats and tilt-in-space mechanisms.

1.1.2 Power Assist Wheelchairs

Power assist wheelchairs are mobility devices designed to provide electric or battery-powered assistance to wheelchair users during propulsion. Unlike powered wheelchairs, which are entirely motorized and typically controlled using joysticks or other interfaces, power-assist wheelchairs are manually propelled by the user and supplemented by motorized assistance. These devices feature motors or other power sources integrated into the wheels or frame of the wheelchair, which augment the user's manual propulsion efforts. Power assist

(a) Standard Powered Wheelchair (b) Mid-Drive Powered Wheelchair (c) Standing Powered Wheelchair

Figure 1: A few powered wheelchair solutions

wheelchairs offer several benefits, including increased range, improved efficiency, enhanced independence, customizable support, compatibility with manual wheelchairs, and portability. Unlike powered wheelchairs, which may require specialized controls and more significant adjustments for users, power assist wheelchairs maintain the manual control and familiarity of traditional manual wheelchairs while providing assistance when needed.

Power assist wheelchairs offer numerous distinct advantages over traditionally powered wheelchairs, significantly enhancing mobility and independence for users. Some of the key features include:

- **Customizable Support:** Power assist systems often feature adjustable settings and control options, allowing users to tailor the level of assistance to their specific needs and preferences.
- **Compatibility with Manual Wheelchairs:** Power assist devices can be attached to most manual wheelchairs, offering a cost-effective solution for users who prefer the familiarity and control of their existing wheelchair.
- **Portability:** Many power assist systems are lightweight and easily detachable, making them convenient for transportation and storage when not in use.
- **Improved Quality of Life:** By reducing physical exertion and expanding mobility, power assist wheelchairs contribute to a higher quality of life for wheelchair users, enabling greater participation in daily activities and social engagements.

(a) SmartDrive MX2

(b) Smoov One

(c) Klaxxon Rear Attachment

(d) Yomper+

Figure 2: A few power Assist wheelchair solutions

1.2 Existing Products

Currently, several power assist solutions are available in the market, including the **SmartDrive MX2** [5] , **Alber Smoov One** [6], **Klaxon Rear Attachment** [2], and **Yomper+** [8]. These solutions share a common approach to power assistance, featuring a dockable powered motor unit with a battery and controller. Users can control these devices using either a joystick or wearable sensors, allowing for seamless integration with their manual wheelchairs. With their intuitive design and versatile functionality, these power assist solutions offer users enhanced mobility and independence in their daily lives. Whether navigating busy streets, traversing uneven terrain, or climbing hills, these devices provide reliable propulsion assistance to users, improving their overall quality of life.

1.2.1 Problem with existing Power Assist Solutions

Currently, all these power assist systems require the user to dismount the wheelchair to attach the system or rely on assistance from someone else. However, a potential solution to this inconvenience is to incorporate a long backward handle that the user can use to attach the power assist system without leaving the wheelchair. One might wonder why the power assist is positioned in such an unconventional manner. The reason lies in the wheelchair's center of gravity, which is situated at the center of the axle. Placing the pushing/pulling force at this point would be the most efficient. However, there are challenges with this approach. While individuals with lower spinal injuries may be able to bend over and attach the power assist using a long handle, those with higher spinal injuries may face difficulty. Bending more than 10 degrees can lead to instability and potential falls for individuals with higher spinal injuries. These challenges highlight the need for innovative solutions in current power assist designs.

2 Approach

2.1 Proposed Solution

Design an automatic docking system for wheelchair power assistance, which would autonomously detect the wheelchair and attach itself to it. The design concept entails incorporating all electronics and sensors directly into the power assistance unit, ensuring compatibility with any existing wheelchair. The automatic docking feature addresses the aforementioned challenges, such as reliance on others for attachment or difficulty in bending at greater angles to facilitate attachment.

2.2 Outline of Project

Below are concise descriptions outlining the major phases accomplished in the project up to this point:

1. **Wheelchair Detection:** Utilize object detection algorithms to identify the wheelchair's presence and location in the environment.
2. **Distance and Orientation Estimation:** Develop algorithms to estimate the distance between the power assist system and the wheelchair, as well as the wheelchair's orientation relative to the power assist.
3. **Localization and Mapping:** Implement localization algorithms to determine the wheelchair's precise position and create a map of the surrounding area.
4. **Prototype Wheeled Robot Design:** Design and construct a prototype wheeled robot equipped with sensors and actuators for testing and deploying path planning algorithms.

Each of the steps is more elaborately explained in the next section (Methods).

3 Method

3.1 Wheelchair Detection

3.1.1 Why Neural Network

To detect the wheelchair in the room, the initial step is to identify the wheelchair amidst various objects in the environment. Simple algorithms like edge detection and convolutions may not be sufficient due to the presence of multiple objects in the background. Therefore, a deep learning network was chosen. YOLO [7] was selected for its simplicity and effectiveness in object detection tasks.

3.1.2 YOLO

YOLO, short for "You Only Look Once," is a state-of-the-art object detection system known for its speed and accuracy. Unlike traditional object detection methods that rely on sliding window approaches or region proposal networks, YOLO frames the object detection task as a single regression problem, directly predicting bounding boxes and class probabilities from full images in a single pass. This approach makes YOLO extremely fast, allowing it to achieve real-time performance on standard hardware. Its usefulness lies in a wide range of applications, including autonomous driving, surveillance, augmented reality, and robotics, where fast and accurate object detection is essential for making informed decisions and taking appropriate actions.

For my project, I opted to use YOLO-v8n for training, as we didn't require a very complex network for detecting just two objects: the pushrim and the front castor wheels.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Figure 3: Various models of YOLO v8

3.1.3 Training

For this specific project, a pretrained YOLO network was utilized. This network had already been trained on the COCO dataset, which contains over 150k images. The advantage of using a pretrained network is that we don't have to learn each and every weight and bias (over 3.2 million parameters) from scratch. It has already learned most of the weights and biases important for feature extraction and classification. Therefore, running it over a much smaller training dataset suffices for our objective. Furthermore, the image size is automatically adjusted in YOLO v8 during training. The network was trained for 25 epochs on the training dataset.

3.1.4 Generating training data

Since there were no existing datasets available for wheelchair wheel detection, I had to source a suitable dataset [4] of wheelchairs and manually segregrate and annotate each image for the wheels and the front castor wheels. In total, 333 images were annotated, encompassing over 2000 instances of the objects. For annotation purposes, I utilized the CVAT tool [3], an open-source online annotation platform. Subsequently, the annotated data was divided into 283 images for training and 50 images for validation, with approximately 15% of the images allocated for validation.

3.2 Distance and Orientation Estimation

Now that our algorithm reliably detects the wheel with sufficient accuracy, we can proceed to develop the algorithm for extracting distance and orientation information of the wheelchair from the captured still image. For image capture and algorithm execution, I am using a **logitech C615 Webcam** [1]. The algorithm's performance relies on the resolution and field of view of the camera. Another crucial assumption the algorithm makes is that the camera is always positioned at the same height as the axle of the wheels.

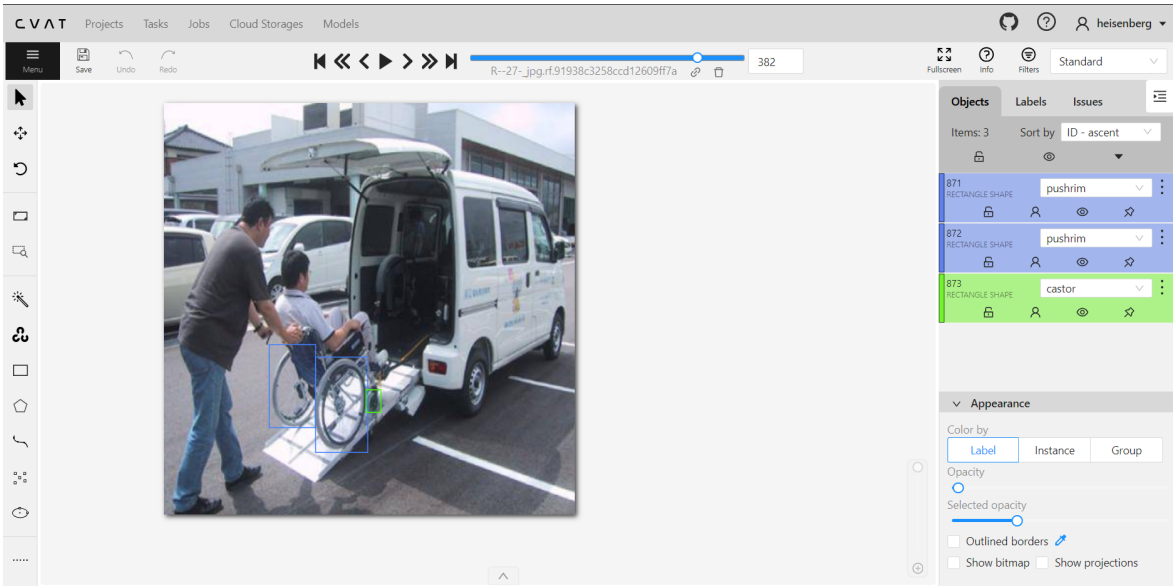


Figure 4: Sample Annotation 1

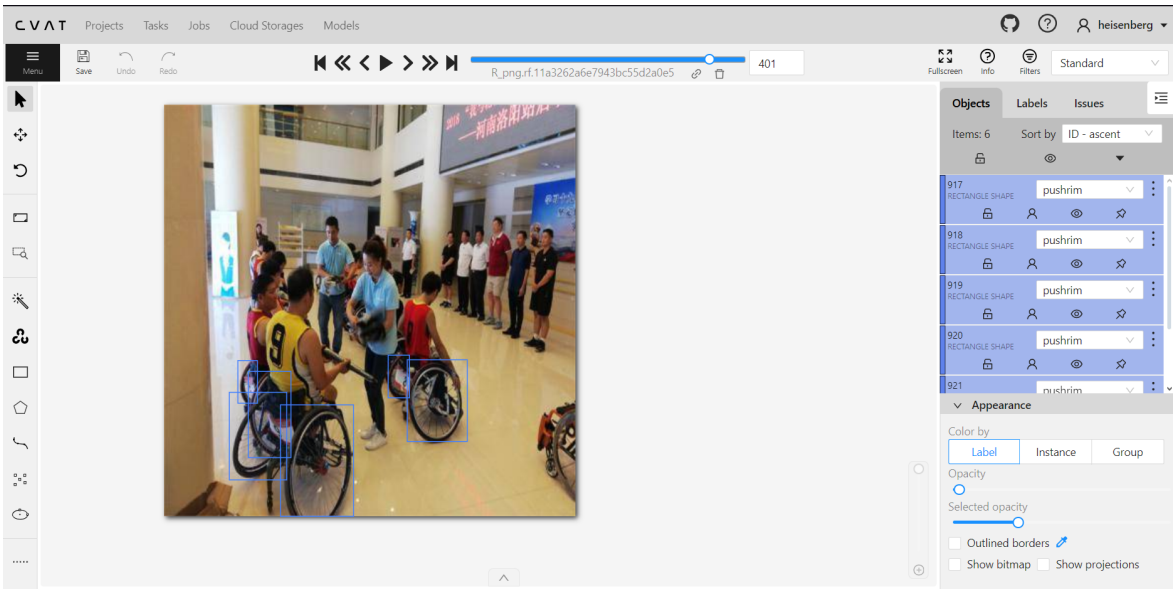


Figure 5: Sample Annotation 2

3.2.1 Distance of wheelchair from camera

Here's how we can determine the distance of the wheel from the camera:

1. To determine distance accurately, we require a ground truth reference in the environment. For our purposes, we've selected the wheelchair's wheel, standardized at 24 inches in diameter, as our reference.
2. An image can be represented as an array, with each pixel containing values for Red, Green, and Blue colors. We've already defined bounding boxes around the wheels, enabling us to count the pixel width and height. We'll use the height as the true value for computation, which I'll explain shortly.

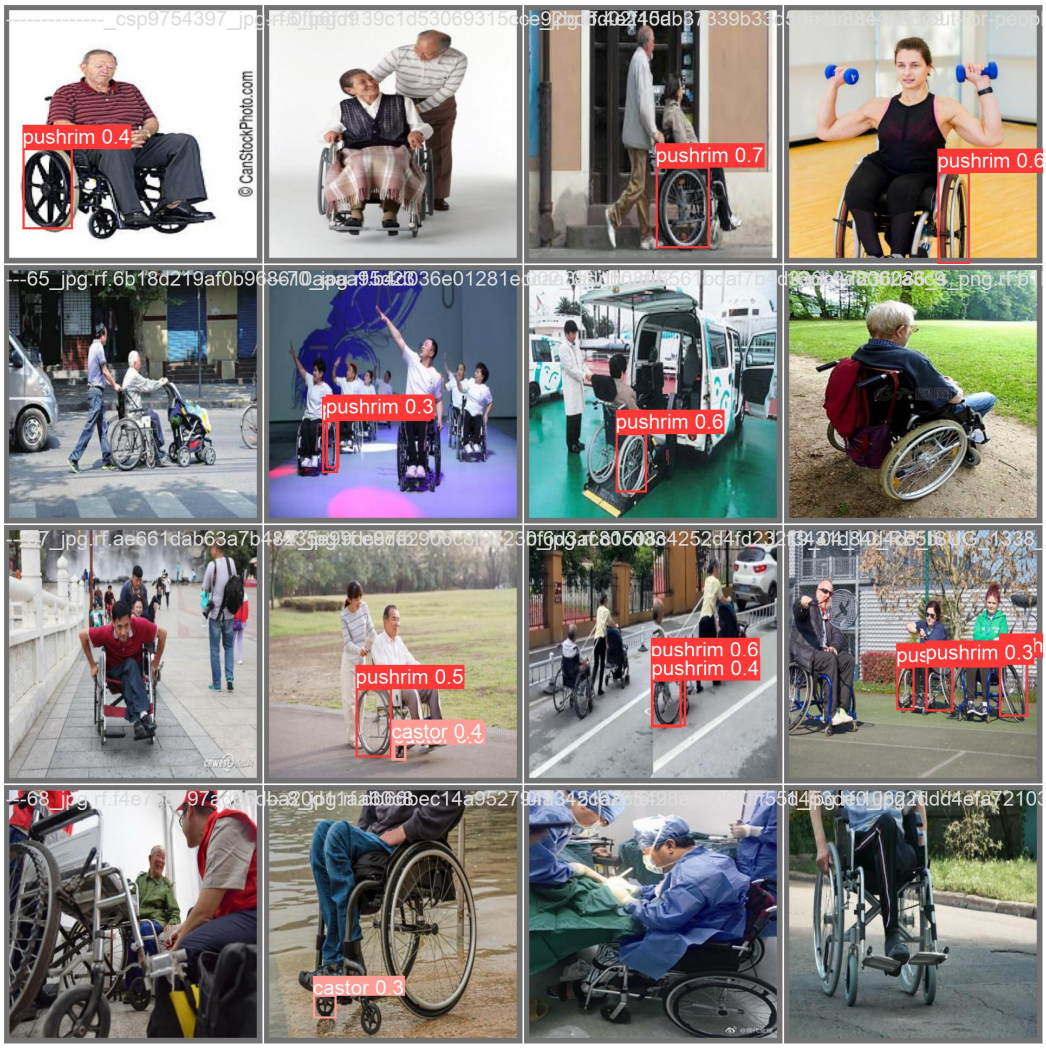
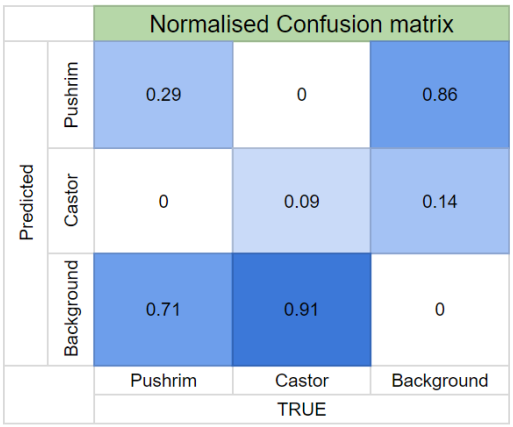


Figure 6: Sample batch validation

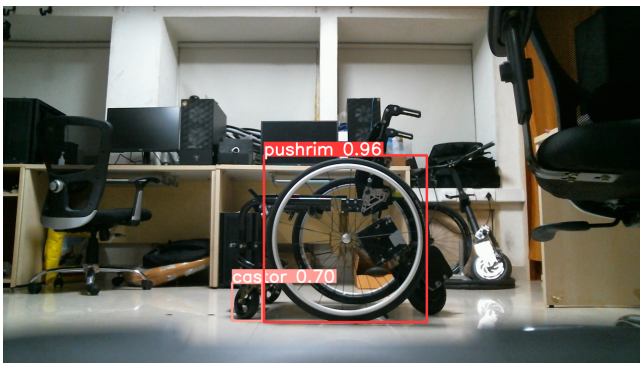


(a) Confusion Matrix

(b) Sample Test Run

Figure 7: Performance of the network

3. We know the wheelchair's height is the diameter, which is 24 inches, corresponding to some fixed pixels in the image. We can employ a simple unitary method to establish the real-world inch-to-pixel ratio on the wheel's plane. Since each pixel forms a square, the same correlation applies to the image's width.
4. We also have information on the total number of pixels along the horizontal axis and the camera's field of view. By taking the arctangent of half the field of view, we obtain the ratio of the horizontal half-length to the distance from the camera.
5. Using these relationships, along with inch-to-metric conversion, we can determine the wheel's distance from the camera. Below are equations and figures to aid in understanding the process.



(a) Test on image with black objects in background

(b) Camera position with respect to wheel

Figure 8: Test Setup

3.2.2 Angle of the Wheelchair

Here's how we can determine the angle of the wheel from the normal of the camera:

1. To estimate the angle the wheel makes from the normal to the camera, we first need to determine the number of pixels the center of the wheel is offset from the centerline.
2. We can obtain the center coordinate from the centroid of the bounding box surrounding the wheel, as both the wheel and the box have nearly coincident centroids.
3. By calculating the offset distance in pixels, we can convert it to real-world distance using the pixel-to-real-world distance conversion factor.
4. Taking the arctangent of the ratio of the offset distance to the distance of the wheelchair from the camera gives us the angle of the wheelchair from the camera's normal.

3.2.3 Orientation of the Wheelchair

1. An image can be interpreted as the 2D projection of the 3D environment. Suppose the wheelchair has some yaw; in that case, the wheel would appear to be squished in the image as it is projected onto a

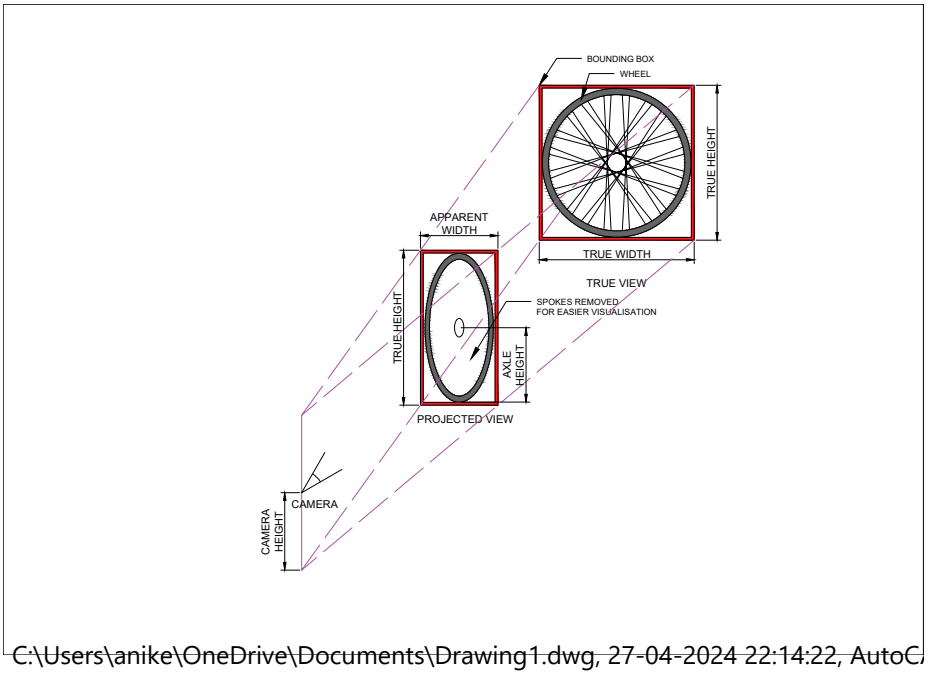


Figure 9: Projected view of the wheel

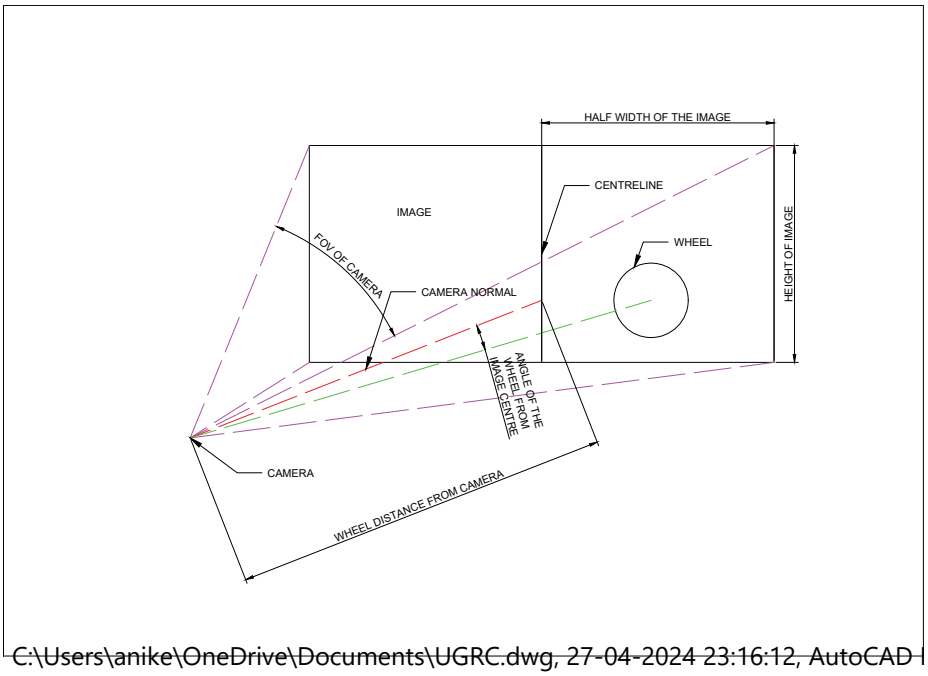
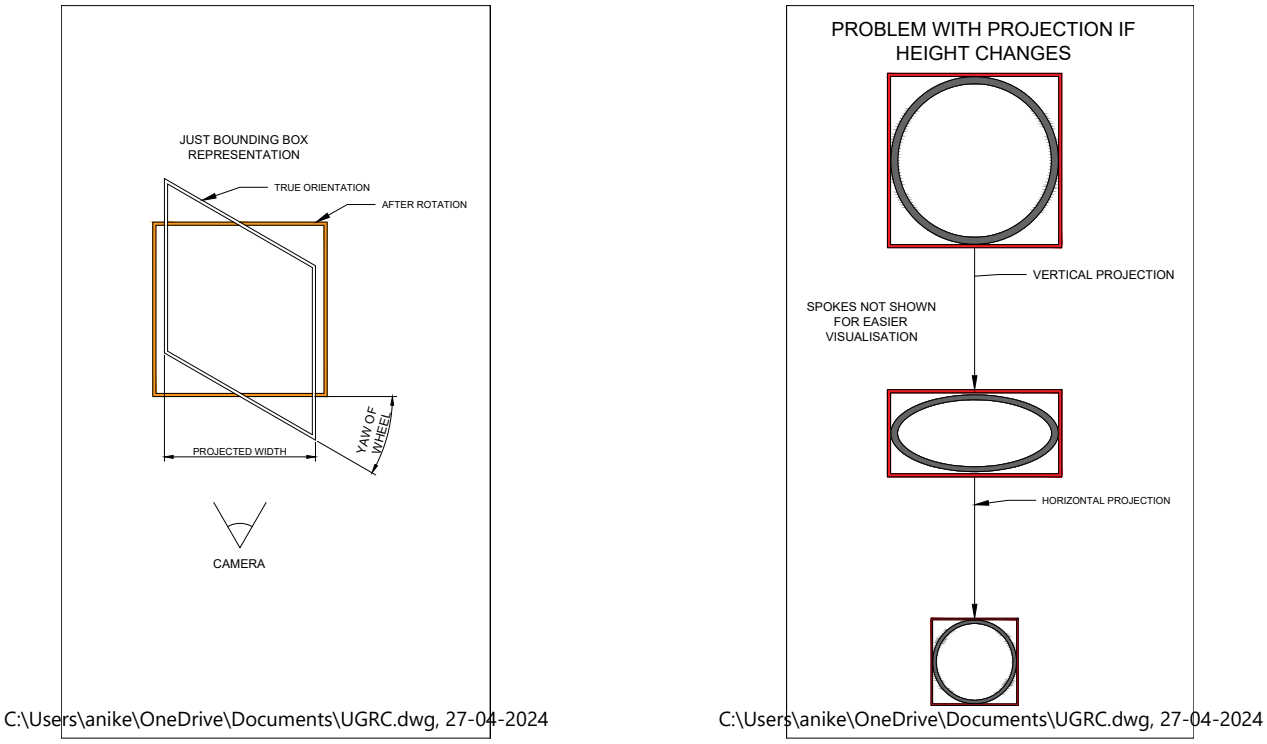


Figure 10: Diagram with the variables

plane normal to the camera's normal.

2. Using simple trigonometry, we can estimate the yaw of the wheelchair. Just taking the cosine inverse of the apparent width of the wheel divided by the real width of the wheel would give us the yaw of the wheelchair.

Note: As mentioned before, the camera must be at the same height as the axle of the wheel to avoid any vertical projection of the wheel. If both horizontal and vertical projections are present, it becomes impossible to determine whether the camera is truly far away from the wheel or if it is horizontally and vertically offset at a closer distance. Therefore, the robot carrying the camera must be designed so that the camera is always at the axle height of the wheel.



(a) Yaw of the wheelchair

(b) Multiple Projections

Figure 11: Illustrations

3.2.4 Equations and Diagrams

Equations:

$$D_{\text{per Pixel}} = \frac{D_{\text{wheel}}}{H_{\text{wheel}}}$$

$$D_{\text{from Camera}} = D_{\text{per Pixel}} \times \frac{W_{\text{image}}}{2} \times \arctan\left(\frac{\alpha}{2}\right)$$

$$\alpha_{\text{wheel}} = \frac{X}{D_{\text{from Camera}}}$$

$$\theta_{\text{wheel}} = \arccos\left(\frac{D_{\text{wheel apparent}}}{D_{\text{wheel}}}\right)$$

Variable Definitions:

$D_{\text{per Pixel}}$: Real-world distance corresponding to 1 pixel

D_{wheel} : Wheel diameter (24 inches)

$D_{\text{wheel apparent}}$: Apparent wheel diameter

H_{wheel} : Number of pixels for the wheel's height

α : Field of view of the camera (68 degrees)

W_{image} : Entire width of the image

$D_{\text{from Camera}}$: Distance from the camera to the wheel

α_{wheel} : Is the angle the wheelchair makes from the normal of the camera

θ_{wheel} : Yaw of the wheelchair

X : Centroid pixel X coordinate from centerline

Note: An important requirement for the algorithm to work accurately is that the camera must be at the same height as the axle of the wheel, as mentioned before. This is necessary because otherwise, the image would appear distorted. Instead of having a projected view for just the width of the wheel, even the height would be a projected view. Since our entire algorithm assumes that the height of the wheel is the true value, all readings would be erroneous if this condition is violated.

3.3 Localization and Mapping

Now that we have obtained the appropriate distance, angle, and orientation of the wheelchair, the next step would be to develop an algorithm to create a map of the environment and define the coordinate system. For our case, we are assuming that the wheelchair is always at the origin, which is (0,0) in the XY plane. Therefore, all the distance and angle data that we are obtaining from the camera data need to be adjusted to localize the camera's position with respect to the wheelchair.

For our experimental purposes, we have assumed an obstacle-free environment. Furthermore, it is a 2D environment since the wheeled robot can only move in terms of X, Y, and yaw motions. Therefore, plotting a 2D plot is sufficient to illustrate the map.

The main problems encountered here are that we cannot distinguish between the left and right halves of the wheelchair, as well as it's difficult to determine whether we are approaching the wheel or moving away from it, as the views would be symmetric.

The latter issue is addressed by utilizing temporal information. Instead of using single pictures to estimate the robot's position, we record videos continuously. By analyzing a series of frames, we can determine if the robot is approaching or moving away from the wheel. However, since the videos can be quite shaky due to the wheeled robot's motion, we need to smooth out the values. Two algorithms that can be used for this purpose are convolution and Polyak averaging.

Convolution is a common technique used in signal processing and image processing to smooth curves by averaging neighboring data points. However, this process can lead to loss of data resolution as it combines information from adjacent points, resulting in a simplified representation of the original signal.

The convolution formula for smoothing curves is:

$$y[n] = \frac{1}{N} \sum_{k=0}^{N-1} x[n-k]$$

where $y[n]$ is the smoothed output, $x[n]$ is the input signal, and N is the size of the convolution kernel.

Polyak averaging is a method used in optimization and reinforcement learning to stabilize training by averaging parameter values over iterations. It helps in smoothing out fluctuations and converging towards a more stable solution.

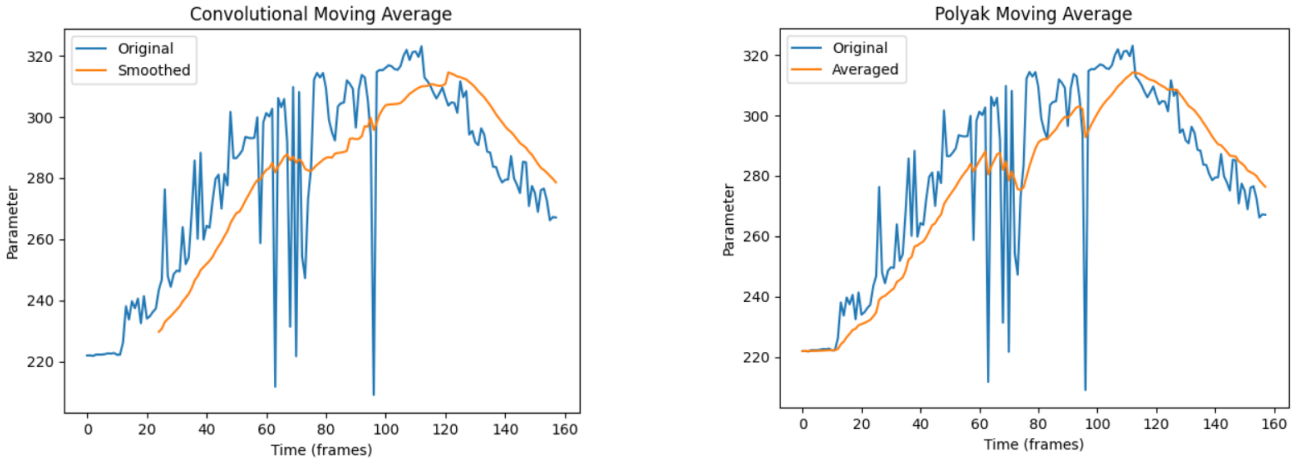
The Polyak averaging formula is:

$$\theta_{\text{avg}} = \alpha\theta_{\text{avg}} + (1-\alpha)\theta_{\text{new}}$$

where:

- θ_{avg} is the averaged parameter vector.
- θ_{new} is the current parameter vector.
- α is the averaging factor, typically a small value close to 1.

Note: Currently, we are assuming that we will develop the path planning algorithm for only one side of the wheelchair. More robust methods for localizing the wheelchair are planned for future stages of the project.



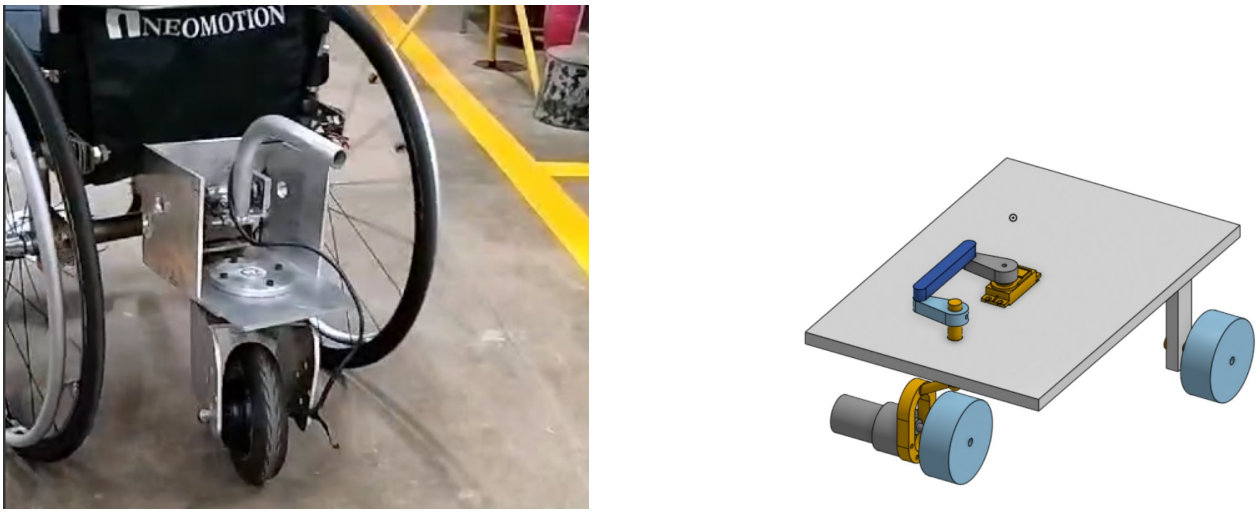
(a) convolution with n = 25

(b) polyak average with $\alpha = 0.9$

Figure 12: Convolution vs Polyak averaging

3.4 Prototype Wheeled Robot Design

A robot, kinematically similar to the power assist, has been designed on the **Onshape** platform to fabricate a prototype power assist and carry out path planning algorithms. Two main types of algorithms are required: closed-loop path planning when the wheel is being detected, and open-loop path planning when the robot is in front of the wheelchair (since the algorithm cannot detect the wheel from that angle very well).



(a) Actual AutoDock

(b) 3D model of Robot

Figure 13: Illustration of the robots

3.4.1 Description of robot

The robot is designed as a 3-wheeled structure, similar to the Autodock robot, with an MG 995 servo controlling the steering of the middle wheel. The two front wheels are meant for stabilizing and are not powered. A 300rpm 12v DC motor is planned to be used for fabricating this robot, and either a Raspberry Pi or Nvidia Jetson will serve as the controller. The flat surface on the robot is intended to provide enough room to mount a tripod with a webcam for recording the wheelchair and performing localization and mapping. The robot is still in the designing phase.

4 Data and Results

4.1 YOLO Performance

The model performs well in accurately detecting wheelchair wheels but struggles with detecting the front view of the wheels and the castor wheels. Possible reasons for the low performance in castor detection include the smaller image size of the castor compared to the entire image, and the castor often being at a peculiar orientation in the dataset, making it appear as a solid dark object. Additionally, the wheelchair wheel detection algorithm sometimes mistakenly identifies bicycle wheels as wheelchair wheels, which is acceptable given their similarity. To improve the model, one could consider training it for more epochs, using a larger dataset with better quality images. However, for our purposes, the current level of accuracy suffices. Furthermore, the algorithm was tested on 10 images, achieving a score exceeding 0.9 for the wheelchair wheel detection each time.

4.2 Orientation detection Performance

The algorithm was able to determine distance, angle and yaw of the wheelchair with relatively high accuracy. If the wheel is centred on the image, then the distance estimates are as accurate as ± 2cm. For testing the accuracy, the camera was fixed at fixed distances of 1m and 1.5m, and then the algorithm was run on it. When the wheel was around the edges of the image, the errors started to increase which is most likely happening due to distortions due to field of view. For the yaw, a similar setup was created and evaluated.

4.3 Localisation and Mapping Performance

To evaluate performance, the wheelchair was recorded from a distance of 1m from the right side, moving from behind to the front. The algorithm effectively mapped the camera's movement relative to the wheelchair with

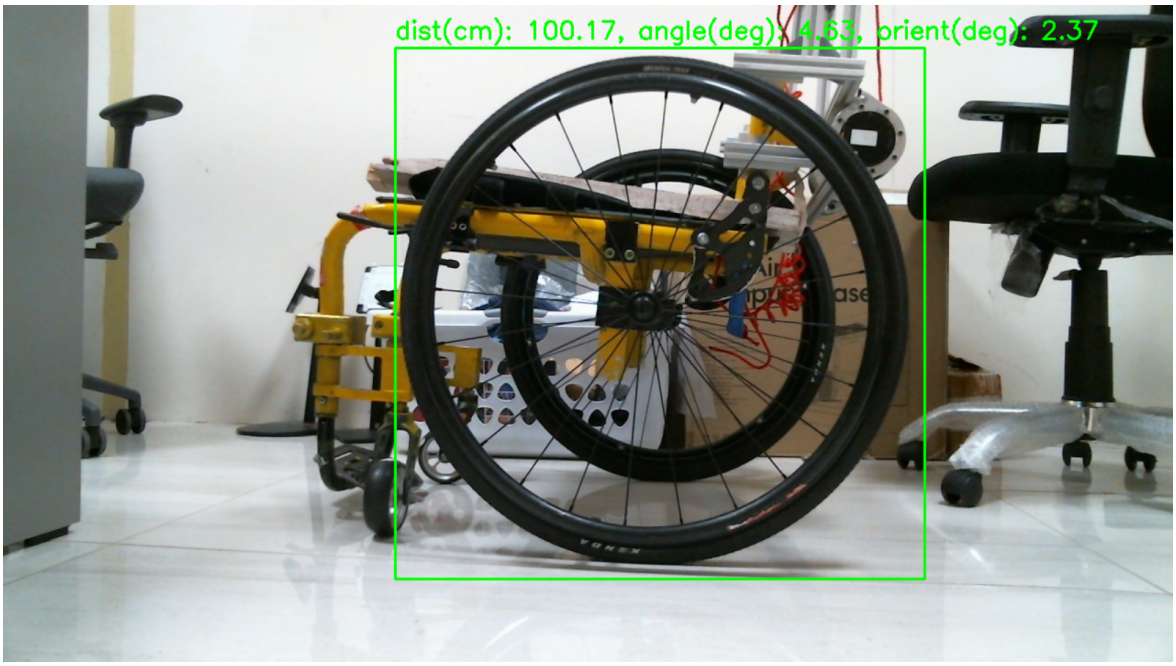


Figure 14: Detection from 1m distance (front view)

Figure 15: Detection from 1.5m distance (front view)

Figure 16: Detection when wheel not centered (1m)

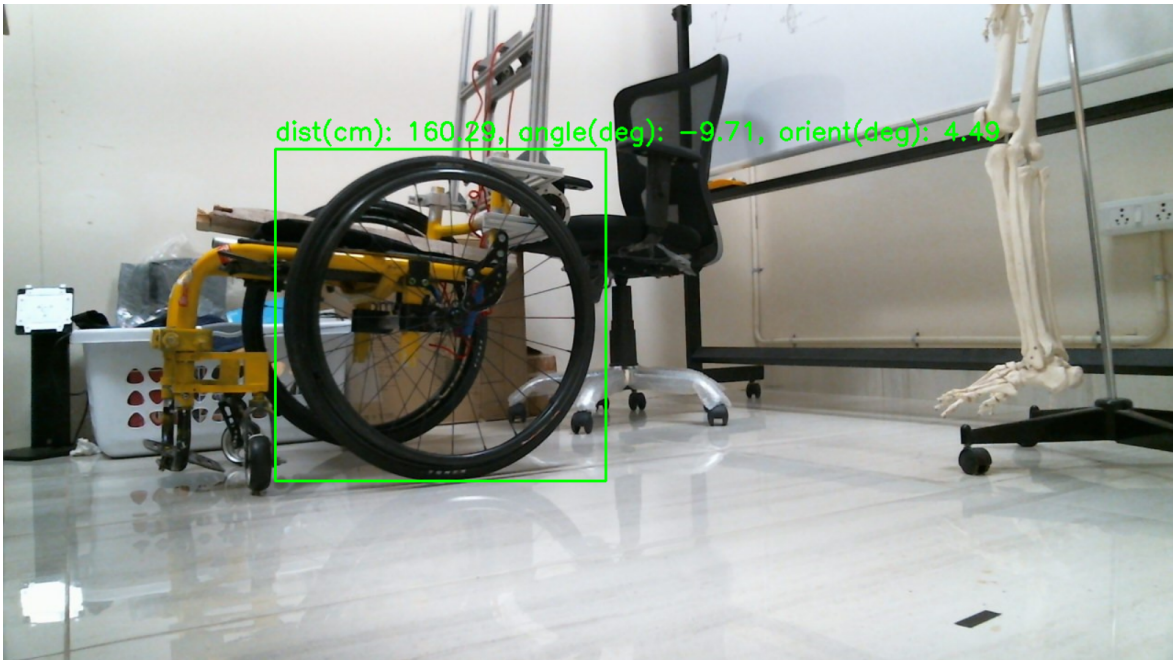


Figure 17: Detection when wheel not centered (1.5m)

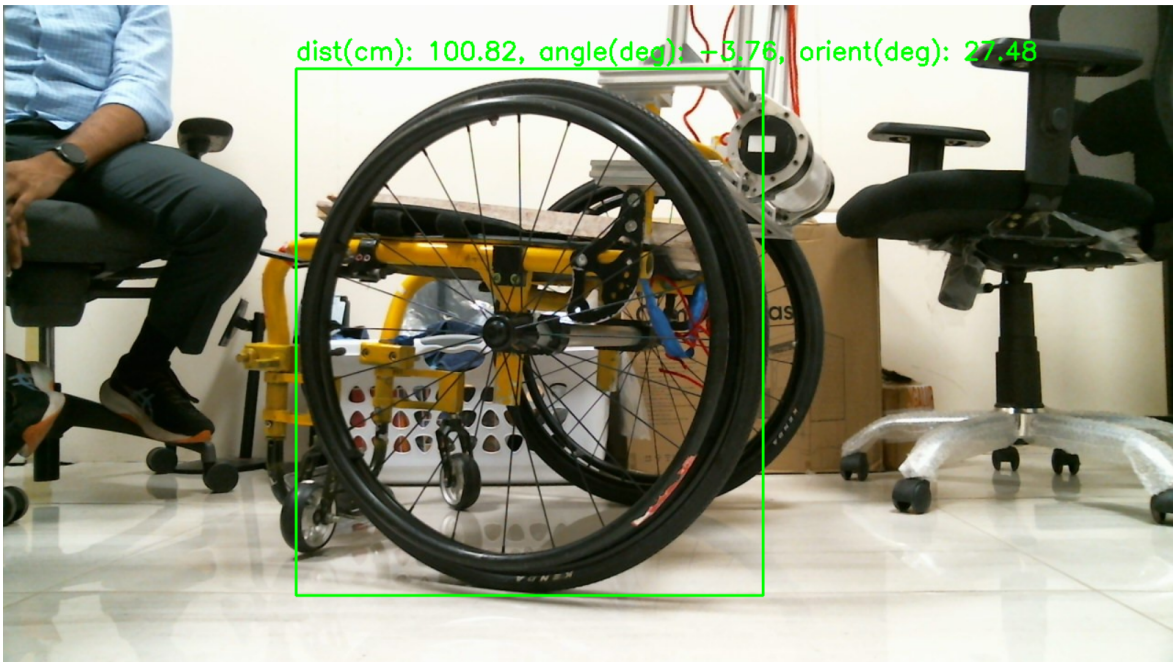


Figure 18: Detection of wheel at 30 degree yaw (1m)

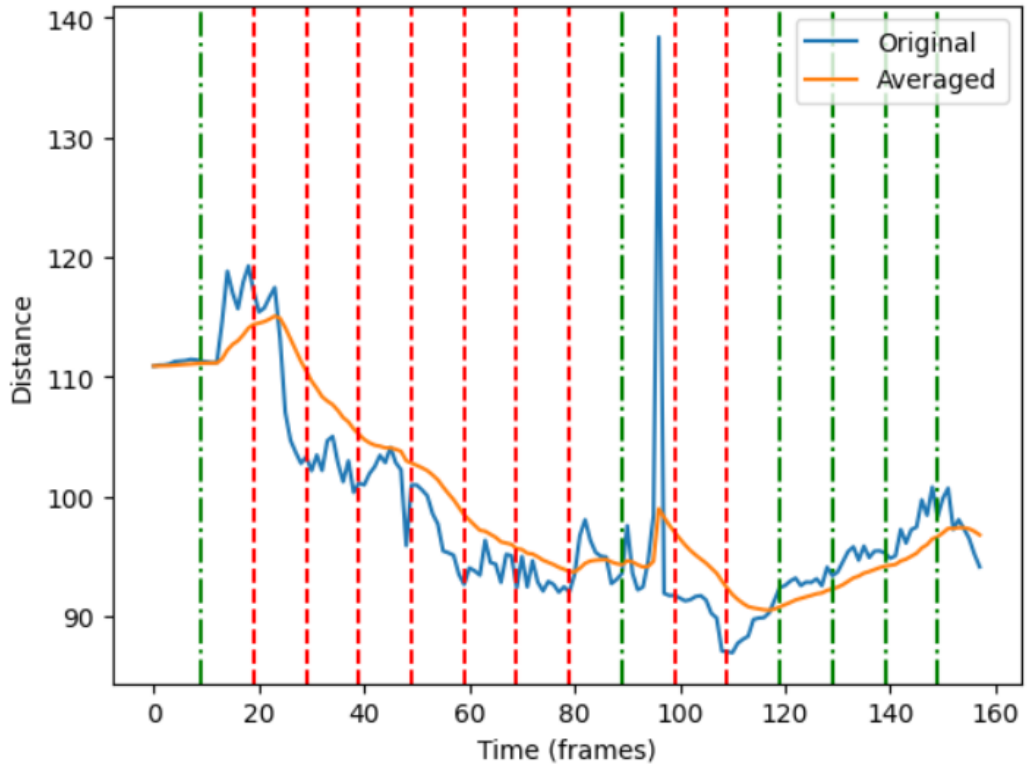


Figure 19: Distances smoothed out, (red: going towards wheelchair, green: going away from wheelchair

relatively good accuracy. It could also discern whether the camera was approaching or moving away from the wheelchair. (**Note:** for the 2d plot, the wheelchair is at the origin, that is (0,0))

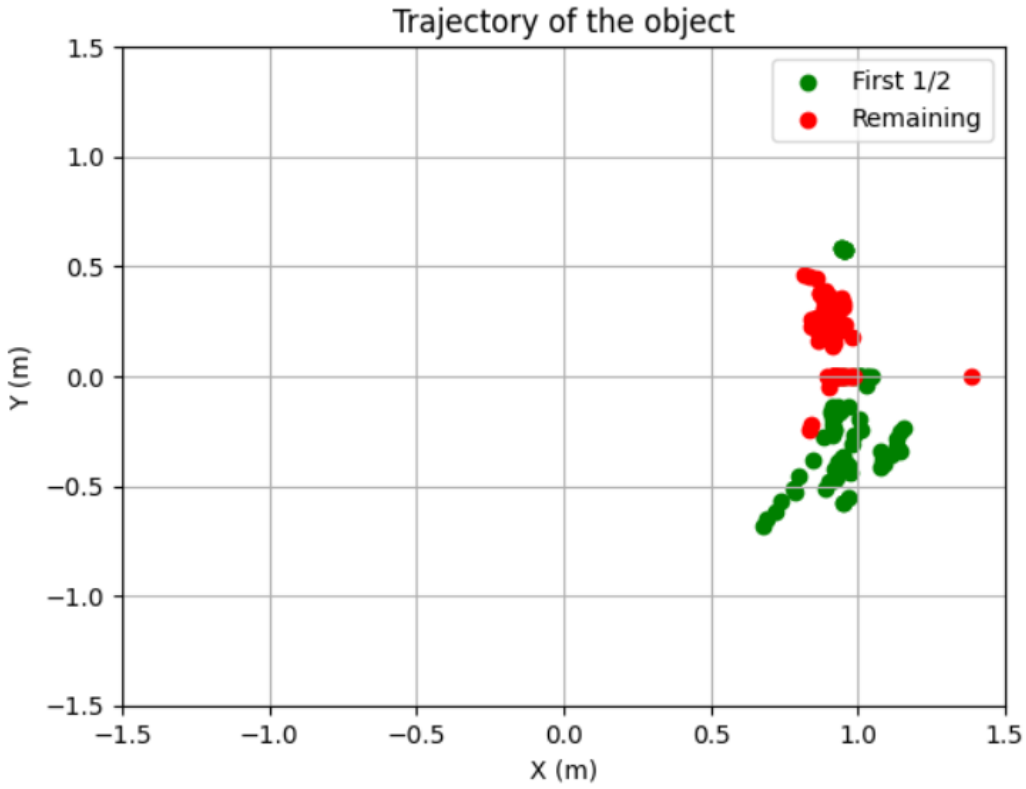


Figure 20: 2D Map with trajectory, going from 4th quadrant to 1st quadrant, wheelchair at (0,0)

5

Discussion

The algorithm performs adequately for our current needs, but there is room for refinement in the future. This could involve acquiring a better training dataset and optimizing the parameters of the neural network.

Since we lacked access to the actual robot, we recorded the wheelchair by manually moving the camera. As a result, the video footage contained considerable shakiness, leading to noisy data for our algorithms. These issues are expected to be resolved when the algorithm is implemented on the actual robot.

Link to the project GitHub Repository: **Autodock**

5.1 Future plans

The next step is to construct a test robot to simulate the motion of the power assist. This robot will serve as a platform for mounting the tripod and conducting path planning algorithms.

Additionally, I am investigating the feasibility of employing CSI-based localization. This approach involves mapping the orientation data of the wheelchair to the individual signal strengths of WiFi signals emitted by the power assist. These signals will reflect off the wheelchair and return, potentially providing valuable

localization information. While ambitious, progress is underway in this area.

References

[1] *Camera*. URL: https://www.logitech.com/en-us/products/webcams/c615-webcam.960-000733.html.

[2] *Klaxon Twist*. URL: https://www.klaxon-klick.com/en/twist/.

[3] *Open Data Annotation Platform*. URL: https://www.cvat.ai.

[4] *Open Source Dataset*. URL: https://universe.roboflow.com/2458761304-qq-com/wheelchair-detection/dataset/1.

[5] *Permobil SmartDrive*. URL: https://hub.permobil.com/smartdrive.

[6] *SMOOV one*. URL: https://smoov.com/us-en/.

[7] *Ultralytics GitHub Page*. URL: https://github.com/ultralytics/ultralytics.

[8] *Yomper Product Page*. URL: https://activerehab.net.au/product/yomper/.