Development of YOLOv8- based Autonomous Wheelchair for obstacle Avoidance

1st Dr. Eko Mulyanto Yuniarno, S.T., M.T.
dept. of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
07211940000073@student.its.ac.id

2nd Dr. Arief Kurniawan, S.T., M.T.
dept. of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
ekomulyanto@ee.its.ac.id

3rd I Gst Ngr Agung Hari Vijaya Kusuma
dept. of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
arifku@ee.its.ac.id

***Ringkasan*—The development of autonomous wheelchairs has become increasingly important in improving mobility for individuals with limited mobility. This study proposes the development of a YOLOv8-based autonomous wheelchair system for obstacle avoidance, specifically focusing on human obstacle detection. By utilizing the advanced object detection capabilities of YOLOv8, the proposed system aims to effectively detect and avoid human obstacles. The system detects humans through video using an Intel NUC and a camera. When an obstacle is detected, the NUC sends a command to the ESP32 to operate the motor to perform avoidance maneuvers. Performance testing of the avoidance success was conducted with 30 trials on stationary human objects. The test results showed that the wheelchair successfully avoided obstacles 30 times, providing a success rate of 100%. This indicates that the designed autonomous wheelchair system is capable of performing obstacle avoidance without mistake.**

***Index Terms*—Autonomous Wheelchair, YOLOv8, Intel NUC, ESP32, Human Detection, Mobility Aid.**

I. Introduction

Mobility is a crucial aspect of daily life that enables individuals to perform various activities independently. However, for individuals who experience paralysis or other mobility impairments, reliance on assistive devices such as wheelchairs becomes inevitable. According to the Indonesian Dictionary (KBBI), paralysis is a condition characterized by the loss of movement function in certain parts of the body, which can be caused by various factors, such as injuries or nerve diseases [1]. Medically, paralysis can occur due to damage to the nervous system, whether central or peripheral, resulting in the loss of motor control in body parts [2].

Technological advancements in this field have significantly contributed to improving the quality of life for people with disabilities. One notable innovation is the development of electric wheelchairs controlled via joystick. Research by Choi, Chung, and Oh demonstrates that integrating motion control of electric wheelchairs with a joystick can enhance user safety and comfort [3]. Nevertheless, challenges remain in terms of navigation and obstacle avoidance, especially in dynamic and complex environments.

With the progress of artificial intelligence technology, deep learning has become a highly effective tool for real-time object detection, identification, and tracking. Lecrosnier's research reveals that employing deep learning in object detection and localization on smart wheelchairs can enhance user mobility and safety in healthcare environments [4]. This technology enables wheelchairs to automatically avoid obstacles, including humans, allowing users to move more safely and efficiently.

The development of autonomous wheelchairs is a vital step towards improving the independence and quality of life for individuals with mobility limitations. With advancements in sensor technology and image processing, the application of object detection methods such as YOLO (You Only Look Once) has become central to innovative autonomous mobility solutions.

In this context, this project aims to enhance the navigation of autonomous wheelchairs by integrating YOLOV8 for obstacle detection and avoidance, ensuring user safety and comfort in various situations.

II. Literature Review

A. Object Detection

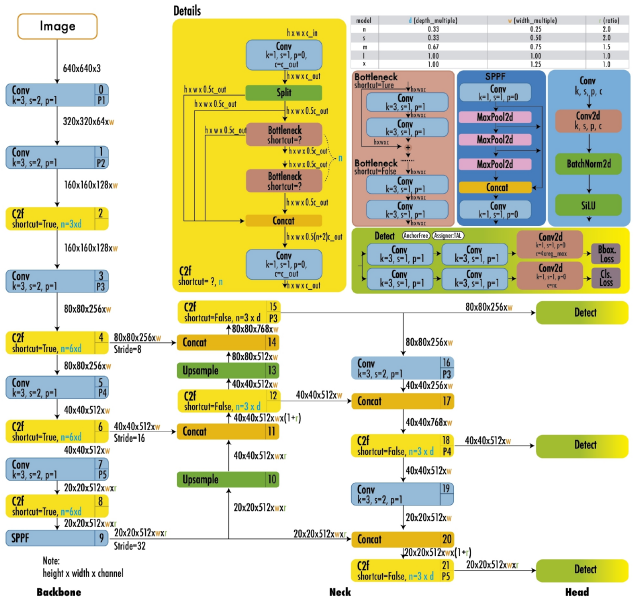
Real-time object detection has emerged as a critical component in various applications, including autonomous vehicles, robotics, video surveillance, and augmented reality. Among various object detection algorithms, the YOLO (*You Only Look Once*) framework stands out for its exceptional balance of speed and accuracy, enabling fast and reliable object identification in images. Since its introduction, the YOLO family has evolved through several iterations, each version building upon the previous to address limitations and improve performance.

B. *YOLO (*You Only Look Once*)*

YOLO, introduced by [5]. Joseph Redmon first introduced the *End to end* approach to real-time object detection. The name YOLO, which stands for *"You Only Look Once"*, refers to its ability to complete the detection task in a single network pass, unlike previous approaches that used sliding windows followed by classifiers that had to be run hundreds or thousands of times per image or more sophisticated methods that divided the task into two steps, where the first step detected possible regions with objects or region proposals, and the second step ran a classifier on those proposals. Additionally, YOLO uses a simpler output based solely on regression to predict detection outputs, as opposed to Fast R-CNN, which uses two separate outputs, a classification for probability, and regression for *box* coordinates [5].

C. *YOLOv8*

YOLOv8 was launched in January 2023 by *Ultralytics*, the company that developed YOLOv5. YOLOv8 provides five scale versions: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra large). YOLOv8 supports various vision tasks such as object detection, segmentation, pose estimation, tracking, and classification [6]. The YoloV8 architecture consists of Convolution 2D layers, C2f (Cross Stage Partial Network), SPPF, Upsampling, and Concate. Figure 1 shows the detailed YOLOv8 architecture.



Gambar 1: YOLOv8 Architecture.

D. *Pose Estimation*

Pose estimation is the task of using machine learning (*ML*) models to estimate a person's pose from an image or video by estimating the spatial locations of key body joints (*keypoints*). Pose estimation refers to a computer vision technique that detects the human figure in images and videos, allowing one to determine, for example, where a person's elbow appears in an image. It is important to recognize that pose estimation only estimates where key body joints are and does not identify who is in the image or video [7].

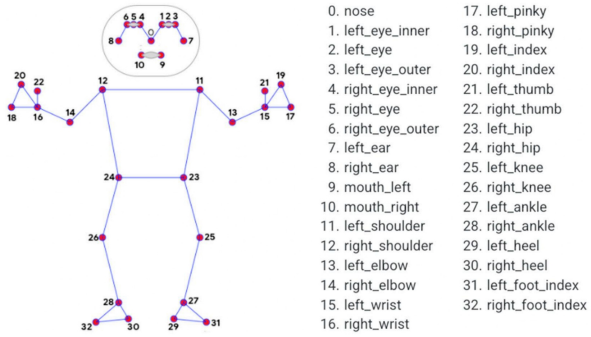
E. *MediaPipe*

MediaPipe is a framework designed by Google for building real-time perception pipelines. MediaPipe allows developers to integrate various types of sensor data, such as video, and other data into one efficient platform that can run on various devices, from mobile to desktop [8].

This framework uses the concept of a "graph" where each node in the graph is a "calculator" that performs specific tasks, such as object detection, pose tracking, or image segmentation. Each of these nodes can be configured through GraphConfig, which describes the topology and functionality of these nodes.

F. *MediaPipe Pose*

MediaPipe Pose (MPP), an open-source cross-platform framework provided by Google, is used to obtain estimates of human joint coordinates in 2D in each image frame. MediaPipe Pose builds pipelines and processes cognitive data in the form of videos using machine learning (ML). MPP uses BlazePose, which extracts 33 2D landmarks on the human body, as shown in Figure 2. BlazePose is a lightweight machine learning architecture that achieves real-time performance on mobile phones and PCs with CPU inference. When using normalized coordinates for pose estimation, the inverse ratio must be multiplied by the y-axis pixel value [9].



Gambar 2: MediaPipe Pose

Precision

Precision is the ratio where TP (true positive) represents the number of true positives and FP (false positive) represents the number of false positives, serving as an evaluation metric in the context of machine learning. It provides a measure of the ratio of correct positive predictions to all positive predictions made by the model. In other words, precision gives insight into how accurately the model makes positive predictions. More specifically, precision reflects how often the model successfully classifies instances as positive within the entire dataset. The precision value can indicate how well the model can provide correct predictions in a positive context. The precision value ranges between 0 and 1, where a value of

1 indicates that all positive predictions made by the model are correct, while a value of 0 indicates that none of the positive predictions are correct.

$$\frac{TP}{TP+FP} \tag{1}$$

The above equation presents the comparison between correct positive predictions and the total positive predictions given by the model, providing a deeper insight into the model's ability to produce accurate results in the desired category.

Recall

Recall is a metric used to measure the ratio of correctly identified positive data to all positive data. Recall provides information about how well a machine learning model finds all positive data. The recall value ranges from 0 to 1. High recall indicates that many positive classes are correctly recognized, or few false negatives are obtained. The formula for recall can be seen in the equation below:

$$\frac{TP}{TP+FN} \tag{2}$$

Recall is the ratio where TP (true positive) represents the number of true positives and FN (false negative) represents the number of false negatives.

Mean Average Precision (mAP)

Mean Average Precision (mAP) is an accuracy metric obtained by calculating the average of the Average Precision (AP). AP itself is obtained through precision and recall calculations. Therefore, mAP can be considered a very informative evaluation metric in assessing a system's performance.

$$AP = \sum((Recall_{n+1} - Recall_n) \times Precision_{interp}) \tag{3}$$

$$\times Recall_{n+1}) \tag{4}$$

$$mAP = \frac{1}{n}\sum_{n}^{i=1} AP_i \tag{5}$$

G. Intersection over Union (IoU)

Intersection over Union, or IoU, is a metric used to evaluate the accuracy of object position detected by a model in image processing. The principle is to calculate the intersection area between the detection box produced by the model and the reference box, which is the gold standard or Ground Truth. This ratio is obtained by comparing the intersection area of the two boxes to the total area they cover together. If we imagine these two boxes as a single unit, IoU gives us a score that measures how well our model predicts the actual object's location. The larger the intersection area relative to the combined total area, the higher the IoU value, indicating better prediction accuracy. Systematically, this is written as:

$$IntersectionoverUnion(IoU) = \frac{|A \bigcap B|}{|A \bigcup B|}. \tag{6}$$

H. Intel NUC

Intel NUC (Next Unit of Computing) is a compact and powerful computing solution designed by Intel to meet various computing needs, from home entertainment to gaming and professional tasks. The Intel NUC features Intel Core processors in a compact 4x4 inch form factor. It is designed to offer the size, performance, sustainability, and reliability needed by modern businesses. Certain Intel NUC models also include Intel vPro® Enterprise technology with enhanced security. This mini PC is upgradable and repairable, making it a versatile choice for various business applications, including client computing, edge computing, and digital signage.

I. ESP32 Devkit V1

The ESP32 Devkit V1 is a development board created by DOIT to run the ESP-WROOM-32 module made by Espressif. The ESP32 Devkit is known as a feature-rich development board with integrated Wi-Fi and Bluetooth connectivity for various applications. This Devkit has many pins that allow it to be programmed with multiple tasks.

J. H-Bridge Motor Driver

An H-Bridge motor driver is an electronic circuit used to control the direction and speed of a DC motor. It works based on four switches forming an H-Bridge, where by controlling the opening and closing of these switches, we can control the direction of the current flowing to the motor. Thus, we can change the rotation direction of the DC motor. The H-Bridge motor driver consists of a set of transistors that function as motor controllers, especially those requiring significant current and voltage. Additionally, the H-Bridge circuit can also provide a braking function to the motor by connecting the two motor terminals, allowing the motor to stop more quickly [10].

K. KY-123 Electric Wheelchair

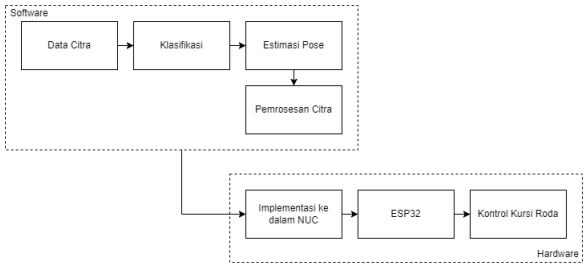
The KY-123 electric wheelchair is a mobility aid that consists of the basic structure of a wheelchair, a motion control system, an electric engine, and a battery module. The advantage of this tool lies in its ability to be easily and comfortably controlled, minimizing the physical effort required by the user compared to a manual wheelchair. This is particularly beneficial for individuals with hemiplegia conditions, allowing operation with one hand. Additionally, this electric wheelchair provides better mobility solutions for the elderly who have limited movement capabilities.

Gambar 3: KY-123 Electric Wheelchair

III. Design and Implementation

A. System Description

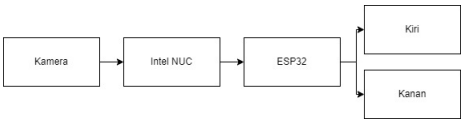
The research and system development are applied according to the design and implementation described in this chapter. This system design includes the concept of creation, design, flow, and infrastructure implementation depicted in a Block Diagram. The design and implementation are illustrated using figures and will be explained from data collection in the form of images, analysis of the model created to detect human objects, and the system using the model as shown in Figure 4 and detailed in each sub-section.



Gambar 4: System Block Diagram

B. Hardware

Hardware design is conducted according to the flow described in this subsection. This design will be presented with a flow block diagram that represents the flow of this hardware design. Figure 5 shows the hardware block diagram as follows:



Gambar 5: Hardware Block Diagram

C. Software

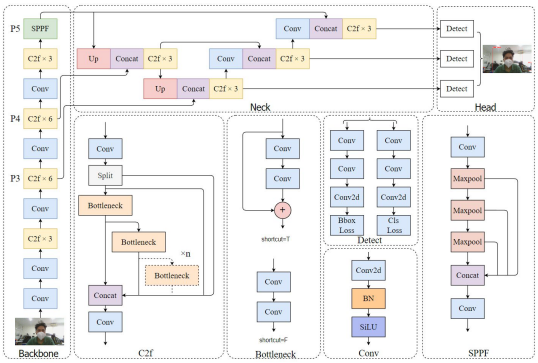
Software design is conducted according to the flow described in this subsection. This design will be presented with a flow block diagram that represents the flow of this software design. Figure 6 shows the software block diagram as follows:

Gambar 6: Software Block Diagram

D. YoloV8 Classification

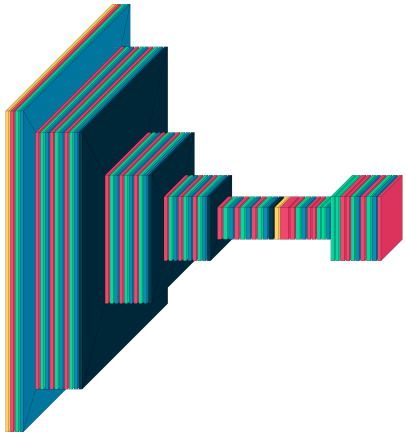
In the classification process, each image that has undergone the labeling process will be recognized using YoloV8, which has been trained to recognize humans in the images. The model will provide output according to the human class, and the classification results will be used as a reference for position in the detection grid.

The model used has an output class of Human, and the values provided by the model include the *Bounding Box* and Confidence Score. The process flow can be seen in Figure 7, which is an architectural block diagram.



Gambar 7: YoloV8 Architecture Visualization

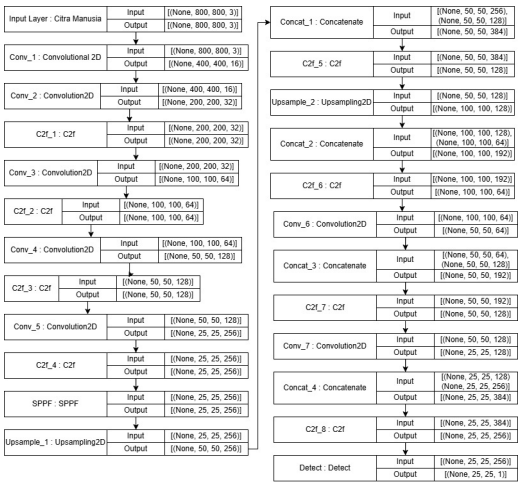
In Figure 8, blue indicates convolutional layers present in the backbone, neck, and head. Yellow indicates C2f residual blocks with convolution. Light green indicates SPPF layers for Spatial Pyramid Pooling Fast. Pink indicates UpSampling layers. Purple indicates Concatenate layers for combining feature maps.



Gambar 8: Visualization with VisualKeras

YoloV8 uses Convolutional Neural Networks (CNN) as the basis of its architecture. In YOLO, CNN is used to extract features from the input image and then apply another network to predict Bounding Boxes and object classes directly from the image.

Based on one of the training sessions, this model has 225 layers, 3011043 parameters, and 8.2 GFLOPs. There are 2D Convolutional layers (Conv2D), C2f blocks, SPPF blocks, UpSampling layers, and Concatenate layers. Figure 9 represents each type of layer with input and output.



Gambar 9: YoloV8 IO Architecture

E. MediaPipe Pose Estimation

In this research, some relevant landmarks are keypoints on the elbows, forearms, and right and left shoulders. These keypoints were chosen based on their visibility and consistency in detection. Table I shows the keypoint numbers and names used in pose estimation.

F. Distance Calculation

One way to calculate distance using a bounding box is through the concept of *focal length pixel* (f_p). Focal length in pixels, or *focal length pixel*, is the conversion of the camera lens's focal length, usually measured in millimeters, into pixels. This is a key concept in photogrammetry and computer vision used to relate visual information from the camera to physical size in the real world.

TABEL I
KEYPOINT TABLE

Keypoint Number	Keypoint Name
11	RIGHT_SHOULDER
12	LEFT_SHOULDER
14	RIGHT_ELBOW
16	RIGHT_WRIST

In the context of this thesis, focal length in pixels is used to convert the obstacle size from pixel units to meter units. This is important because the autonomous wheelchair needs to understand the actual distance to obstacles to make accurate navigation decisions. The formula can be seen in Equation 7:

$$f_p = \frac{D \times h_b}{H_o} \tag{7}$$

Where:

- f_p is the focal length in pixels,
- D is the actual distance from the camera to the object,
- h_b is the height of the bounding box in pixels,
- H_o is the actual height of the object.

When calculating the distance to an object, the detected bounding box height by YOLO is used, combined with the known actual height of the object and the focal length in pixels. This forms Equation 8 as follows:

$$D = \frac{f_p \times H_o}{h_b} \tag{8}$$

Here:

- H_o is the average height of a human or other identified object.

Another approach in determining distance using pose is through the Euclidean Distance method. Euclidean Distance in pixels is a method to measure the straight-line distance between two points in image space, usually measured in pixels. In the context of this thesis, which involves an autonomous wheelchair with MediaPipe integration, this measurement is crucial for various functions, especially in pose analysis and assessing the proportion of objects in the images produced by the camera. The formula can be seen in Equation 9

$$d_p = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \times s_f \tag{9}$$

Where:

- d_p is the Euclidean distance in pixels,
- (x_1, y_1) and (x_2, y_2) are the coordinates of two points measured in pixels,
- s_f is the scale factor.

This formula yields the distance between two points in the same units as the coordinates x_1, y_1, and x_2, y_2. Typically, if these coordinates are measured in pixels, the resulting distance will also be in pixels.

From the above calculations, the distance is still in pixel units. In the context of distance calculation, a standard value in meters is needed to align with general measurement standards. To convert these pixel values into meters, calibration using the value *K* is necessary. The value *K* is a calibration value based on experimental measurements. Equation 10 is as follows:

$$D_m = \left(\frac{K}{d_p} \right) \tag{10}$$

Where:

- D_m is the distance in meters,
- K is the calibration constant,
- d_p is the distance in pixels.

The value *K* determines how much influence the pixel distance has on the meter distance. Larger or smaller values will directly affect the distance calculation results. For example, a larger *K* value will result in a smaller distance for the same number of pixels. This value is used in a very specific context where this parameter describes the direct relationship between pixel size and actual distance or dimensions, based on specific assumptions about the scene's geometry and camera characteristics.

This value must be accurately calibrated to match the specific characteristics of the camera and setup used. Incorrect calibration will result in inaccurate distance measurements, which can impact the autonomous wheelchair's navigation decisions. The calibration formula for the *K* value using Equation 11 is as follows:

$$K = J_o \times U_p \tag{11}$$

Where:

- J_o is the actual object distance,
- U_p is the object size in pixels.

In the calculation of obstacle width, the Euclidean distance calculation is used but with a different application context. Both approaches differ in application and context. The differences can be seen in Equation 12

$$l_p = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \times s_f \tag{12}$$

The output value obtained is the width of the human in pixels. To map the width from pixels to meters, a conversion formula is needed to determine the size in pixels (which is a digital and relative size) to real-world size (meters). The formula can be seen in Equation 13:

$$l_m = l_p \times s_f \tag{13}$$

By multiplying the object's width in pixels by the scale factor, the result is the object's width in meters. This formula is useful in applications where the real-world dimensions of objects need to be known to make accurate decisions or measurements.

The scale factor is a value that converts the size from pixel units to meter units. This value is obtained through the calibration process. The scale factor determines how many meters are represented by each pixel in the image, based on the camera's distance to the object and other camera settings such as focal length. Additionally, note that the scale factor used in this formula differs from the distance formula previously explained. The calibration formula for the scale factor in the context of object width in Equation 14 is as follows:

$$s_f = \frac{D_n}{U_p} \tag{14}$$

Where:

- D_n is the average actual dimension,
- U_p is the average size in pixels.

In the calculation of obstacle width, the Euclidean distance calculation is used but with a different application context. Both approaches differ in application and context. The differences can be seen in Equation 15

$$l_p = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \times s_f \tag{15}$$

The output value obtained is the width of the human in pixels. To map the width from pixels to meters, a conversion formula is needed to determine the size in pixels (which is a digital and relative size) to real-world size (meters). The formula can be seen in Equation 16:

$$l_m = l_p \times s_f \tag{16}$$

By multiplying the object's width in pixels by the scale factor, the result is the object's width in meters. This formula is useful in applications where the real-world dimensions of objects need to be known to make accurate decisions or measurements.

The scale factor is a value that converts the size from pixel units to meter units. This value is obtained through the calibration process. The scale factor determines how many meters are represented by each pixel in the image, based on the camera's distance to the object and other camera settings such as focal length. Additionally, note that the scale factor used in this formula differs from the distance formula previously explained. The calibration formula for the scale factor in the context of object width in Equation 17 is as follows:

$$s_f = \frac{D_n}{U_p} \tag{17}$$

Where:

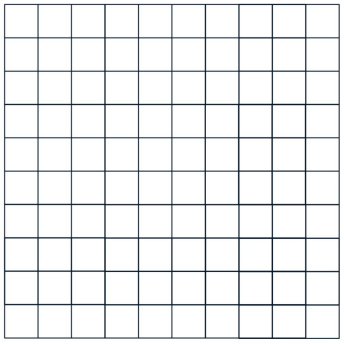
- D_n is the average actual dimension,
- U_p is the average size in pixels.

These values can be used in the code to convert sizes from pixels to real sizes based on calibrated measurements. Calibration must be done correctly to obtain accurate results.

G. Grid Implementation

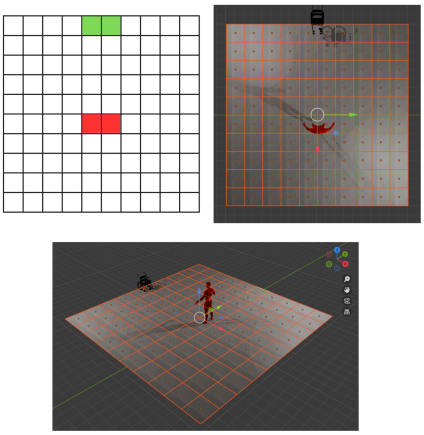
In this thesis, the autonomous wheelchair must be able to determine the position of obstacles it will encounter. Therefore, a map is needed as a reference for the wheelchair to take action based on the obstacle distance and width, which will be used to determine where the wheelchair should avoid and whether the obstacle has been successfully avoided. Grid is one of the best approaches in mapping detection results. Using a grid not only facilitates decision-making but also provides easy-to-understand visualization. The grid size can also be adjusted according to needs, both in dimensions and appearance.

After the above formulas are implemented in the code, several important variables will be obtained to be used in mapping the position and size of obstacles in the grid. Before that, here is the grid used, as shown in Figure 10:



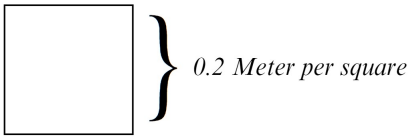
Gambar 10: 10x10 Grid

The grid is created using the OpenCV library. This library is chosen because of its easy and lightweight implementation. For better understanding, the explanation will be accompanied by a 3D visualization using Blender in Figure 11.



Gambar 11: Grid Visualization with Blender

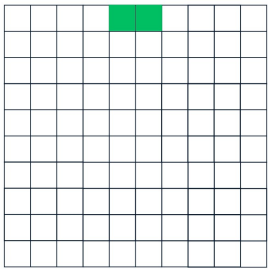
The use of a 10x10 grid is based on image results and model performance. Both bounding boxes and poses have limitations where their calculation results do not exceed the parameters set on the grid. These parameters can be seen in Figure 12:



Gambar 12: Parameters for each grid box

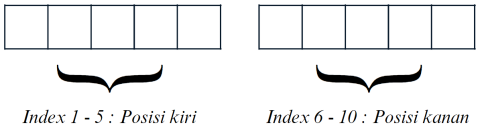
From the above image, each box in the grid is 0.2 meters in both vertical and horizontal positions. The determination of 0.2 meters per box is based on calibration and experimental measurements ensuring this size is optimal for accurate and efficient obstacle detection.

To map objects well in the grid, an index must be created to determine the object's relative left and right positions to the camera. For 10 horizontal boxes, they will be divided into left and right indices, where indices 1 to 5 are categorized as left indices, and indices 6 to 10 are categorized as right indices. This decision is related to the static wheelchair position on the grid, depicted in Figure 13:



Gambar 13: Wheelchair Position on Grid

It can be seen that the wheelchair position occupies 2 green grid boxes at (5,1) and (6,1). This decision is based on the wheelchair's width, which fits the grid size. The top position determines which indices are left and right on the grid. With the top position set as the wheelchair's constant position and the input image results obtained as mirrored detection results, the indices can be depicted according to Figure 14:



Gambar 14: Position Categories based on Index

This categorization plays an important role in the wheelchair's turn decision-making, where the detection results in the form of distance, width, and relative position will be displayed on the grid.

H. Wheelchair Avoidance Navigation

1) Results without Detection: In this condition, no objects are detected, meaning there are no obstacles blocking the wheelchair's path, so the wheelchair will continue to move forward. In this condition, the bounding box, pose, grid, and others are not displayed because no humans are detected. It can be seen in the example image 15

Gambar 15: Example condition without Human Detection

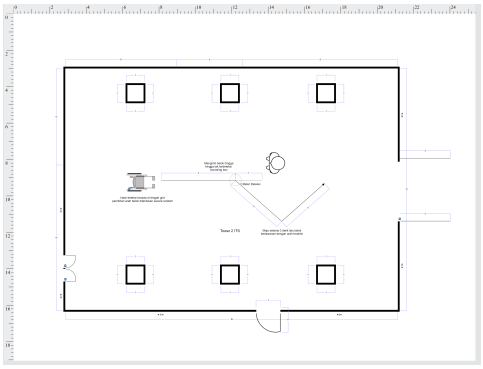
It can be seen in Figure 15, there is text "No Humans Detected," which means no human objects are detected.

2) Object detection results on the grid showing Left Index greater than Right Index: In this condition, the detection result position shows a higher value on the left index (indices 1-5), meaning the object is on the left side relative to the wheelchair position. This can be seen in the example image below.

Gambar 16: Example condition Left Index >Right

It can be seen in Figure 16, the grid value is more towards the left index than the right. Based on this position, the wheelchair will avoid to the right, which is a safer turn than to the left.

The wheelchair can only be said to avoid if it returns to its original direction. Thus, Figure 17 is the avoidance scheme that will be followed in this condition.



Gambar 17: Avoidance scheme in condition Left Index >Right

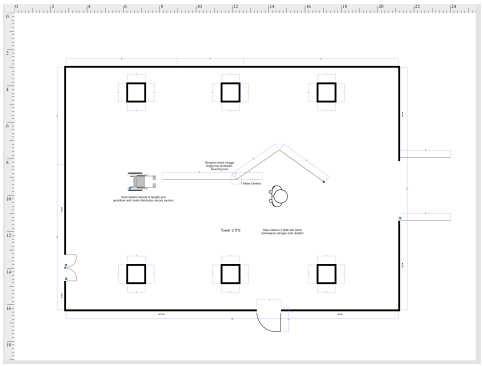
It can be seen in Figure 17 that when the wheelchair is 1 meter from detection, it will turn according to the index condition and store this turn direction. If the bounding box is no longer visible, the wheelchair will move forward for 4 seconds, then check the last turn direction. Then the wheelchair will turn again according to the last direction for 5 seconds, then reset the last direction condition. Finally, in this condition, the wheelchair will enter the no detection condition, so it will continue to move forward until the next detection.

3) Object detection results on the grid showing the right index greater than the left index: In this condition, the detection result position shows a higher value on the right index (6-10), meaning the object is on the right side relative to the wheelchair position. This can be seen in the example image 18

Gambar 18: Example condition Right Index >Left

It can be seen in Figure 18, the grid value is more towards the right index than the left. Based on this position, the wheelchair will avoid to the left, which is a safer turn than to the right.

The wheelchair can only be said to avoid if it returns to its original direction. Thus, Figure 19 is the avoidance scheme that will be followed in this condition.



Gambar 19: Avoidance scheme in condition Right Index >Left

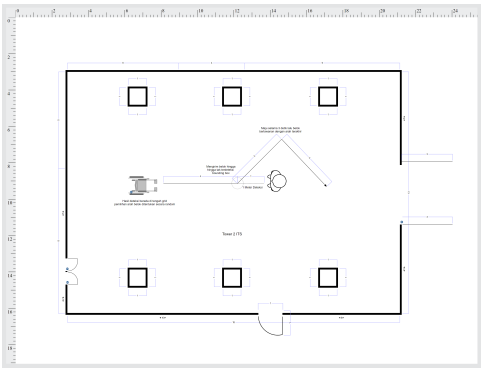
It can be seen in Figure 19 that when the wheelchair is 1 meter from detection, it will turn according to the index condition and store this turn direction. If the bounding box is no longer visible, the wheelchair will move forward for 4 seconds, then check the last turn direction. Then the wheelchair will turn again according to the last direction for 5 seconds, then reset the last direction condition. Finally, in this condition, the wheelchair will enter the no detection condition, so it will continue to move forward until the next detection.

4) Object detection results on the grid showing a linear position relative to the wheelchair: In this condition, a command must be added for decision-making where the turn position in this condition, either to the right or left, will not have advantages/disadvantages because in this position the index value is equal on both the right and left. Therefore, a good approach must be made so that this decision-making does not cause errors. In this project, decision-making is based on a random value. Thus, the decision taken can be either to the right or left.

Gambar 20: Example condition Linear

It can be seen in Figure 20, the grid is aligned with the wheelchair position. Thus, the use of random values will be very useful in making decisions when facing such cases.

The wheelchair can only be said to avoid if it returns to its original direction. Thus, Figure 21 is the avoidance scheme that will be followed in this condition.



Gambar 21: Avoidance scheme in condition Linear

It can be seen in Figure 21 that when the wheelchair is 1 meter from detection, it will turn according to the index condition and store this turn direction. If the bounding box is no longer visible, the wheelchair will move forward for 4 seconds, then check the last turn direction. Then the wheelchair will turn again according to the last direction for 5 seconds, then reset the last direction condition. Finally, in this condition, the wheelchair will enter the no detection condition, so it will continue to move forward until the next detection.

IV. Results and Discussion

A. FPS Testing

As shown in Table II on the right, the average FPS value in the laptop FPS test is 13.140. The highest FPS value is 13.23, and the lowest FPS is 13.05. Additionally, as seen in Table II, the average FPS value in the Intel NUC test is 6.111. The highest FPS value is 6.47, and the lowest FPS is 5.54.

Tabel II
FPS Comparison Results on Laptop and NUC

	Laptop	NUC
Average FPS	13.14	6.11
Maximum FPS	13.23	6.47
Minimum FPS	13.05	5.54

B. Response Time Testing

Based on the above output, the system's Response Time can be calculated and will be explained in Table III. The Response Time will be tested to obtain the time required for detection with the model, classification, and transmission to the ESP32 until the wheelchair motor starts moving. This test is conducted in real-time on the NUC device, with delay calculations obtained from the start of transmission until the motor stops. The inference time calculation starts from the beginning of the prediction process until the classification result is obtained. The average delay time obtained is 0.2494 seconds from the NUC test data, and the results can be seen in the table below. The average inference time obtained is 139.4899 ms or 0.1394899 seconds.

TABEL III
DELAY RESULTS

	per second
Average Delay	0.249
Maximum Delay	0.379
Minimum Delay	0.145

TABEL IV
INFERENCE RESULTS

	per millisecond
Average Inference	139.489
Maximum Inference	181.100
Minimum Inference	123.100

C. Detection Distance Accuracy Testing

This test evaluates the model's ability to generate distances based on calculations on the *Bounding Box* and pose. The test compares the actual object distance with the system-generated distance on an Intel NUC against a standing human. Calibration was performed at a distance of 150 cm, chosen for pose and bounding box visibility. The resulting Focal Length is 480, with K1 and K2 values of 10.922 and 24.222, respectively. These values will be used in distance accuracy tests at 150 cm, 100 cm, and 50 cm. The test aims to evaluate the system's distance measurement capability.

The test was conducted using a measuring tape attached to the camera and extended towards the researcher to obtain the distance. The values were calculated to obtain the average *difference* or discrepancy produced by the system against actual measurements. The following table summarizes the average distance differences for each test.

TABEL V
SUMMARY OF DETECTION DISTANCE ACCURACY TEST RESULTS

Distance	*Yolo Bbox*	*MediaPipe Shoulder*	*MediaPipe Hand*
150 cm	3.2 cm	5.06 cm	23.8 cm
100 cm	20.8 cm	2.2 cm	3.53 cm
50 cm	69.8 cm	14.8 cm	1.93 cm

In Table V, it is shown that at a distance of 150 cm, the largest error is in the hand landmark with a percentage of 15.86%, and the smallest error is in the Yolo Bbox at 2.13%. At a distance of 100 cm, the largest error is in the Yolo Bbox with a percentage of 20.8%, and the smallest is in the shoulder landmark with a percentage of 2.2%. At a distance of 50 cm, the largest error is in the Yolo Bbox with a percentage of 139%, and the smallest is in the hand landmark with a percentage of 3.86%.

TABEL VI
OBSTACLE AVOIDANCE SUCCESS PERFORMANCE TABLE

Trial	Result
1	Wheelchair Successfully Avoided
2	Wheelchair Successfully Avoided
3	Wheelchair Successfully Avoided
4	Wheelchair Successfully Avoided
5	Wheelchair Successfully Avoided
6	Wheelchair Successfully Avoided
7	Wheelchair Successfully Avoided
8	Wheelchair Successfully Avoided
9	Wheelchair Successfully Avoided
10	Wheelchair Successfully Avoided
11	Wheelchair Successfully Avoided
12	Wheelchair Successfully Avoided
13	Wheelchair Successfully Avoided
14	Wheelchair Successfully Avoided
15	Wheelchair Successfully Avoided
16	Wheelchair Successfully Avoided
17	Wheelchair Successfully Avoided
18	Wheelchair Successfully Avoided
19	Wheelchair Successfully Avoided
20	Wheelchair Successfully Avoided
21	Wheelchair Successfully Avoided
22	Wheelchair Successfully Avoided
23	Wheelchair Successfully Avoided
24	Wheelchair Successfully Avoided
25	Wheelchair Successfully Avoided
26	Wheelchair Successfully Avoided
27	Wheelchair Successfully Avoided
28	Wheelchair Successfully Avoided
29	Wheelchair Successfully Avoided
30	Wheelchair Successfully Avoided

In Table VI, the results show that avoidance was successful 30 times. Therefore, the success rate obtained from this test is 100%. This result demonstrates that the system is capable of detecting and avoiding humans effectively.

D. Avoidance Accuracy Performance

The distance avoidance accuracy test measures the comparison between the system-generated avoidance distance and the real-world wheelchair avoidance distance from a human. The avoidance distance is set at 100 cm or 1 meter, meaning the wheelchair must avoid at this distance if a human is detected.

TABEL VII
SUMMARY OF AVOIDANCE ACCURACY PERFORMANCE RESULTS

Distance set	*Real Error*	*System Error*
100 cm	33.1 cm	29.3 cm

As shown in Table VII, the average real measurement error is 33.1 cm, and the average system measurement error is 29.3 cm. The results do not match the set distance of 100 cm or 1 meter and tend to decrease in accuracy.

This decrease is caused by several factors, including the camera's position shaking during testing, system delays, the laptop not being charged, limiting GPU usage, and decreased laptop performance during testing due to increased laptop heat over time, resulting in lower FPS.

E. Avoidance Success Performance with Two Obstacles

TABEL VIII
AVOIDANCE SUCCESS PERFORMANCE TABLE WITH TWO OBSTACLES

Trial	Result
1	Wheelchair Successfully Avoided
2	Wheelchair Successfully Avoided
3	Wheelchair Successfully Avoided
4	Wheelchair Successfully Avoided
5	Wheelchair Successfully Avoided
6	Wheelchair Successfully Avoided
7	Wheelchair Successfully Avoided
8	Wheelchair Successfully Avoided
9	Wheelchair Successfully Avoided
10	Wheelchair Successfully Avoided

The results show that avoidance was successful 10 times. Therefore, the success rate obtained from this test is 100%. This result demonstrates that the system can detect and avoid two human obstacles effectively.

F. Avoidance Success Performance with Three Obstacles

TABEL IX
AVOIDANCE SUCCESS PERFORMANCE TABLE WITH THREE OBSTACLES

Trial	Result
1	Wheelchair Successfully Avoided
2	Wheelchair Successfully Avoided
3	Wheelchair Successfully Avoided
4	Wheelchair Successfully Avoided
5	Wheelchair Successfully Avoided

The results show that avoidance was successful 5 times. Therefore, the success rate obtained from this test is 100%. This result demonstrates that the system can detect and avoid three human obstacles effectively.

G. Avoidance Success Performance with Four Obstacles

TABEL X
AVOIDANCE SUCCESS PERFORMANCE TABLE WITH FOUR OBSTACLES

Trial	Result
1	Wheelchair Successfully Avoided
2	Wheelchair Successfully Avoided
3	Wheelchair Successfully Avoided

The results show that avoidance was successful 3 times. Therefore, the success rate obtained from this test is 100%. This result demonstrates that the system can detect and avoid four human obstacles effectively.

V. CONCLUSION

Based on the testing results, the following conclusions can be drawn:

1) The model with the highest metrics trained with various configurations is the one with the highest mAP score at IoU 0.5 of 81.85%. This value is sufficient for performing avoidance, as seen from the excellent avoidance performance results.
2) The performance of the NUC in FPS testing produced a lower value compared to the author's personal laptop, with a difference of 7.029 fps.
3) The average delay obtained in the testing was approximately 0.2494 seconds, and the average inference value obtained was 139.4899 ms or 0.1394 seconds.
4) The results show that detection using *Bounding Box* and shoulder landmarks is more accurate at longer distances (150 cm and 100 cm), while arm landmarks are more accurate at closer distances (50 cm). The best average *difference* for the bounding box at 150 cm is 3.2 cm, the best average *difference* for shoulder landmarks at 100 cm is 2.2 cm, and the best average *difference* for arm landmarks at 50 cm is 1.93 cm.
5) Detection performance results were satisfactory in 30 test samples, with a success rate of 100%, indicating that the system created can avoid humans very well.

VI. SUGGESTIONS

For further development in future research, the following suggestions can be given:

1) Increase the variety of datasets to enhance detection performance.
2) Use a device with better performance for higher fps.
3) Improve the detection grid performance by making more detailed adjustments for better mapping.
4) Use a device cooler when testing in an open room to avoid performance degradation.

PUSTAKA

[1] K. Daring, "Kbbi vi daring," 2016. [Online]. Available: https://kbbi.kemdikbud.go.id/entri/lumpuh

[2] P. Pansawira, "Kelumpuhan - gejala, penyebab, dan mengobati - alodokter," 4 2022. [Online]. Available: https://www.alodokter.com/kelumpuhan#:~:text=Kondisi%20ini%20dapat%20disebabkan%20oleh,Penanganan%20kelumpuhan%20tergantung%20pada%20penyebabnya.

[3] J. H. Choi, Y. Chung, and S. Oh, "Motion control of joystick interfaced electric wheelchair for improvement of safety and riding comfort," *Mechatronics*, vol. 59, pp. 104–114, 2019.

[4] L. Lecrosnier, R. Khemmar, N. Ragot, B. Decoux, R. Rossi, N. Kefi, and J.-Y. Ertaud, "Deep learning-based object detection, localisation and tracking for smart wheelchair healthcare mobility," *International journal of environmental research and public health*, vol. 18, no. 1, p. 91, 2021.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[6] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, p. 1680–1716, Nov. 2023. [Online]. Available: http://dx.doi.org/10.3390/make5040083

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[8] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C. Chang, M. G. Yong, J. Lee, W. Chang, W. Hua, M. Georg, and M. Grundmann, "Mediapipe: A framework for building perception pipelines," *CoRR*, vol. abs/1906.08172, 2019. [Online]. Available: http://arxiv.org/abs/1906.08172

[9] J.-W. Kim, J.-Y. Choi, E.-J. Ha, and J.-H. Choi, "Human pose estimation using mediapipe pose and optimization method based on a humanoid model," *Applied Sciences*, vol. 13, no. 4, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/4/2700

[10] Muhammad, "Analisis driver h-bridge menggunakan relay pada motor bldc konstruksi axial flux (celah ganda)," 2018. [Online]. Available: https://repository.unej.ac.id/bitstream/handle/123456789/89217/Muhammad%20-%20161910201115.pdf?sequence=1&isAllowed=y