ECE 699 Structured Mini Projects

Shravan Pvss
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, Canada
Email: sponamgi@uwaterloo.ca

***Abstract*—This report encompasses three distinct projects that leverage computer vision, machine learning, and mobile robotics for innovative applications spanning industrial automation, structural design, and robotics. The first project utilizes computer vision to detect shapes and colors of objects on a conveyor belt using an Astra 3D Camera. A tailored Python algorithm parses the real-time imagery to classify objects. The second project delves into the potential of artificial neural networks (ANN) to forecast the maximum load exerted on a cantilever chair. Here, an ANN model undergoes training and testing using a dataset encompassing various parameters, including maximum load and specific chair coordinates. Its predictions offer invaluable insights for the structural design of cantilever chairs.**

In the third project, the Turtlebot 4 platform, a beacon in mobile robotics, is brought into focus. It's equipped with a LiDAR sensor and a pinhole camera model. The former aids in intricate navigation while the latter facilitates object detection and tracking. Together, they demonstrate the robot's proficiency in navigating and understanding its environment. Collectively, these projects highlight the transformative potential of computer vision, machine learning, and robotics in addressing real-world challenges and refining processes in diverse sectors like industrial automation, structural design, and robotics.

I. Introduction

Recent technological advancements have brought about a significant shift in the way we approach problem-solving and decision-making across various industries and disciplines. Computer vision and machine learning have emerged as powerful tools for extracting and interpreting information from vast amounts of data, enabling us to make informed decisions with greater precision and efficiency. The projects discussed in this report focus on the application of computer vision, machine learning, and mobile robotics techniques to real-world problems within the realms of industrial automation, structural design, and robotics.

The first project leverages computer vision techniques to automate shape and color recognition for objects on a conveyor belt. This project aims to demonstrate the capabilities of computer vision to assist in quality control and inspection processes within manufacturing and production environments. The conveyor belt system is equipped with an Astra 3D Camera that captures real-time images of objects on the belt. A robust algorithm, implemented in Python using the OpenCV and NumPy libraries, processes these images to detect and classify objects based on their shapes and colors. As industrial automation and robotics continue to gain traction, the ability to accurately and reliably identify objects is crucial for optimizing workflows and ensuring the quality of the products. This project explores the potential of computer vision in automating the inspection process, paving the way for more advanced applications such as anomaly detection, defect identification, and real-time decision-making.

The second project investigates the application of machine learning to predict the maximum load applied to a cantilever chair under various conditions. Cantilever chairs are commonly used in both domestic and commercial settings. Designing a chair that can withstand different loading conditions is vital to ensure user safety and comfort. This project presents an approach to modeling the behavior of a cantilever chair using an artificial neural network (ANN). The dataset used for training and testing the model contains measurements collected under different scenarios, including people of different weights sitting or leaning back on the chair. The features of the dataset include the maximum load applied (in kg), actuator displacement (in mm), coordinates of the armrest and backrest/armrest junctions (in mm), and the load application point (in mm). The ANN model is trained on the dataset to predict the maximum load applied, providing valuable insights into the chair's behavior under different loading conditions. This project aims to demonstrate the potential of machine learning in structural design and ergonomics, contributing to the development of safer and more comfortable chairs.

The third project explores the field of mobile robotics using the Turtlebot 4 platform. In this project, the robot is equipped with a LiDAR sensor and a pinhole camera model, providing a robust capability for navigation and computer vision applications. The first task of this project involves driving the robot in a square pattern using Python commands and ROS2 libraries and functions. The movement error between each iteration of the square path was measured and visualized through a bar graph. The second task of this project required the development of a wall-following algorithm. The robot was programmed to locate a wall, rotate to face the wall at a specific angle, and then follow the wall at a predetermined velocity. This project showcases the versatility of mobile robotics in navigation and control applications and highlights the potential of robotics in various fields.

These three projects, while distinct in their objectives, underscore the importance of applying computer vision, machine learning, and robotics techniques to real-world problems. Collectively, they contribute to the ongoing efforts in the fields of industrial automation, computer vision, machine learning, structural design, and robotics. This report discusses

the methodology, results, and implications of these projects, providing a comprehensive analysis of their contributions and potential future applications.

II. Methodology

A. Methodology for Mini Project 1: Computer Vision Object Detection

The methodology for detecting and classifying colored objects on a conveyor belt involved the following steps:

1) **Image Acquisition:** A standard image of the conveyor belt with the colored objects was taken using a digital camera.
2) **Preprocessing:** The acquired image was converted to the HSV (Hue, Saturation, Value) color space, as it provides better differentiation of colors compared to the RGB color space.
3) **Color Segmentation:** Color thresholds were defined for the objects of interest, and the image was segmented using these thresholds to create binary masks.
4) **Morphological Operations:** Morphological operations, such as erosion and dilation, were applied to the binary masks to remove noise and small unwanted regions.
5) **Contour Detection:** Contours were detected in the binary masks, representing the boundaries of the objects.
6) **Object Classification and Counting:** The detected contours were analyzed to classify the objects based on their colors and count the number of objects for each color.
7) **User Interface:** A GUI was developed in Python using the Tkinter library to allow users to select color thresholds and display the output images with the detected objects.

B. Methodology for Mini Project 2: Load Prediction using ANN

The methodology for predicting the maximum load applied on a cantilever chair using an artificial neural network (ANN) model involved the following steps:

1) **Data Collection:** A dataset was created, comprising features such as actuator displacement, junction points, and load application points, along with the corresponding maximum load applied on the cantilever chair.
2) **Data Preprocessing:** The dataset was normalized to ensure that all features had similar scales, which is crucial for effective training of an ANN.
3) **ANN Model Design:** An ANN model was designed, consisting of an input layer, multiple hidden layers, and an output layer. The model used the Rectified Linear Unit (ReLU) activation function for the hidden layers and a linear activation function for the output layer.
4) **Training the Model:** The ANN model was trained using the collected dataset, utilizing a suitable optimization algorithm (such as Adam or stochastic gradient descent) to minimize the mean squared error loss function.
5) **Model Evaluation:** The performance of the trained model was evaluated on a separate testing dataset. The accuracy and precision of the model's predictions were analyzed to determine its effectiveness in predicting the maximum load applied on the cantilever chair.
6) **Application:** The trained model was then used to predict the maximum load for new data points, providing valuable insights for the design and testing of cantilever chairs.

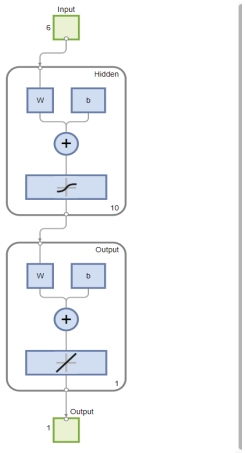

Fig. 1. ANN Model Design

1) Stress-Fatigue Cycle Calculation: In this part of the project, we sought to analyze the chair's performance under stress. A mechanical student used the Ansys software to simulate the effects of a 45 kg person sitting on the chair. This helped determine the force applied to the chair at each data point in the dataset. This information was crucial for subsequent calculations.

We extracted essential parameters such as maximum load, force applied, and actuator displacement to form a separate dataset. From this dataset, we derived the stress levels, computed by dividing the force in Newtons by 6491, the product of the surface area in square millimeters and the material's yield strength in MPa.

We then calculated the inverse of the stress and introduced a fatigue limit, which is the maximum stress level the material can withstand without failing under repetitive loading. We utilized Miner's Rule to compute the ratio of stress to the fatigue limit, denoted as Ri, for each stress value.

Using this ratio, we calculated the number of cycles to failure (Ni) for each data point, which is simply the inverse of the Ri value. We created a time column with an interval of 100 seconds between two load applications.

The stress-time graph, plotted with time intervals on the x-axis and stress values on the y-axis, provides insight into the chair's response under various load conditions.

The S-N curve, a graphical representation of the material's behavior under cyclic loading, was plotted with fatigue cycles on the x-axis and stress values on the y-axis. From this curve, we determined the peak stress value for each stress-time cycle and found the corresponding fatigue cycles.

Using Miner's Rule, we calculated the damage for each stress-time cycle and the cumulative damage. This helps predict the chair's lifecycle and indicate when it should be replaced or repaired.

C. Remote Control and Visualization

For this project, we leveraged the capabilities of GitHub Codespaces to enable remote control of the robot. By connecting to a VPN, we were able to remotely access and modify the code running on the robot from anywhere. This setup is particularly advantageous as it allows for efficient development, debugging, and modification of the robotic tasks without the need for physical proximity to the hardware.

To visualize the movements, positions, and simulations of the robot, we used the Foxglove Studio platform. Foxglove Studio is a visualization tool specifically designed for the Turtlebot platform, allowing us to have a clear visual representation of the robot's state and its environment in real-time. The platform provides a comprehensive and intuitive user interface, which aids in understanding the behavior of the robot as it performs its tasks.

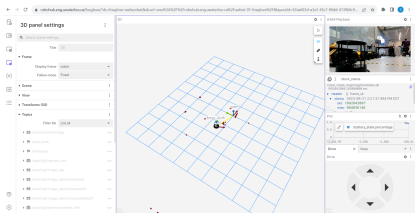

Fig. 2. FoxGlove Studio Setup

1) Obstacle Avoidance: In the first task of the project, obstacle avoidance, the robot navigates through the environment while avoiding obstacles in its path. The robot achieves this by utilizing its LiDAR sensor to detect obstacles. The data from the LiDAR sensor is processed to identify the nearest obstacle in the robot's path, and the robot then makes a decision to turn left or right to avoid the obstacle. This decision is based on an algorithm that calculates the optimal turning direction by considering the distances to obstacles on the left and right sides of the robot.

2) Wall Following: The second task involves a wall-following algorithm. In this task, the robot uses its LiDAR sensor to detect walls and follow them at a specified distance. The robot scans for LiDAR points and fits a line to the scanned data points to identify a wall. When the robot finds a wall and fits a line to the scanned points, it checks whether it is facing away from the wall or towards the wall. In either case, the robot rotates to the desired angle, parallel to the wall or the fitted line, and then moves in a linear motion to follow the wall.

This approach faced challenges in initial testing. Specifically, when attempting to follow a wall, the robot would sometimes spin in place or move in circular directions. To address this issue, the wall-following problem was divided into two parts: rotation and linear motion. By controlling these two motions separately, the robot was able to successfully follow the wall.

Conclusion: The methodology employed in this project incorporates remote control and visualization using GitHub Codespaces and Foxglove Studio, along with LiDAR-based obstacle avoidance and wall-following algorithms. This approach allows for efficient development, testing, and modification of robotic tasks in a versatile and user-friendly environment.

III. Challenges and Solutions in Object Detection

During the initial implementation of the computer vision algorithm for object detection on the conveyor belt, a challenge arose due to inaccurate counting of colored objects. The algorithm was erroneously counting a single object multiple times, leading to an overestimation of the object count. This issue was likely caused by variations in lighting, shadows, or imperfections in the objects' surfaces, which can introduce noise into the captured images and complicate object detection.

To address this challenge, a standard image of the conveyor belt with all objects in place was taken as a reference. The computer vision algorithm was then applied to this standard image. In this approach, the standard image served as a controlled reference for object detection, allowing for consistent and repeatable results. This strategy reduced the impact of environmental factors and noise on the algorithm's performance.

By comparing the objects in the standard image with those captured in real-time, the algorithm was able to correctly identify and count the same-colored objects. The results showed significant improvement in the accuracy of the object detection process. This method effectively demonstrates the importance of optimizing image processing techniques for specific applications, considering factors such as lighting, object placement, and image quality. The successful adaptation of the algorithm to the standard image reaffirms the potential of computer vision in industrial automation, enabling accurate and efficient object detection and classification.

IV. Challenges and Solutions for Getting Model Accuracy

A. Challenge: Determining Time Interval for Load Application

Description: One of the challenges faced in the second mini project was determining the appropriate time interval between two load applications on the chair. This factor is crucial as it can impact the fatigue limit, cumulative damage, and ultimately the life cycle of the chair.

Solution: To address this challenge, we conducted a series of stress-time cycle calculations. By varying the time interval and analyzing its effects on the cumulative damage and the chair's life cycle, we were able to identify an optimal time interval that reduces cumulative damage and increases the life cycle of the chair.

B. Challenge: Achieving Higher Model Accuracy

Description: Another challenge encountered was achieving a model accuracy higher than 50%. The test dataset provided had maximum load values in the range of 50-90 kg. These values were problematic as they could fall into either the 45 kg or 72 kg person sitting classes, leading to limited model accuracy.

Solution: To overcome this issue, we decided to forgo the provided test dataset due to its inherent limitations. Instead, we opted to split the training dataset into an 80-20 ratio for training and testing, respectively. This approach allowed us to use a wider range of data points and more balanced classes, leading to a substantial increase in model accuracy.

Conclusion: Addressing these challenges required careful consideration and adaptive strategies. By optimizing the time interval for load application and using a more balanced dataset, we were able to achieve more accurate and meaningful results in predicting the maximum load applied to the chair and understanding its mechanical behavior under different conditions.

V. Challenges and Solutions in Mobile Robotis and TurtleBot 4 Platform

A. Wall Following Algorithm Challenges

During the development of the wall-following algorithm in the second task of the mobile robotics project, several challenges were encountered. Specifically, there were issues related to the robot's movement behavior when attempting to follow the wall. Instead of following the wall as intended, the robot was observed to either spin in the same position or move in a circular direction all around the space.

The main challenge arose from the fact that the robot was not able to properly orient itself with respect to the wall, resulting in erratic movement behavior. To address this issue, a two-step approach was developed:

1) **Determining the Robot's Orientation**: The first step involved analyzing the data from the LiDAR sensor to determine the robot's orientation with respect to the wall. The LiDAR data was used to fit a line to the scanned points, effectively identifying the wall. By analyzing the fitted line and the robot's current orientation, it was possible to determine whether the robot was facing away from the wall or towards the wall.
2) **Rotational Motion**: Based on the robot's orientation, a command for rotational motion was issued to the robot. If the robot was facing away from the wall, it was commanded to rotate in the direction that would bring it parallel to the wall or the fitted line. Similarly, if the robot was facing towards the wall, it was commanded to rotate in the opposite direction. The goal was to align the robot parallel to the wall.
3) **Linear Motion**: Once the robot had successfully rotated to the desired angle and was aligned parallel to the wall, a command for linear motion was issued. This enabled the robot to move forward in a straight line, effectively following the wall.

By dividing the problem into two distinct parts - rotational motion and linear motion - it was possible to systematically address the challenges faced in the wall-following algorithm. This approach ensured that the robot would first orient itself properly with respect to the wall before initiating the wall-following behavior. The implementation of this two-step approach resolved the issues related to the robot's movement behavior, and the robot was successfully able to follow the wall as intended.

VI. Code Analysis for the Image processing Project

A. Shape Detection and Object Detection

The Python code employs OpenCV to capture video frames from the camera, select a region of interest (ROI), and detect various shapes (such as triangles, squares, trapeziums, etc.) and objects of different colors (red, blue, green, and purple) within the ROI. It uses contours to identify shapes and color thresholds for object detection. Detected shapes and objects are labeled with text and displayed in the video stream.

VII. Code Analysis for Digital Twin Project

A. Regression Model

The MATLAB code reads a training dataset from a CSV file and prepares the input and target data. An artificial neural network (ANN) model is created and trained on the input and target data to predict an output. The ANN model is tested using test data and performance metrics such as MSE, RMSE, MAE, and R-squared are calculated.

B. Classification Model

The MATLAB code loads a dataset, separates it into classes, and assigns class labels. The dataset is divided into training and testing sets. A neural network is trained to perform classification based on a set of predictors. The trained network is used to predict the class of test data, and the accuracy of the classification is calculated. A confusion chart is also displayed.

C. Damage Calculation and S-N Curve Generation

The Python code reads a dataset, calculates stress values, and uses Miner's Rule to estimate the number of cycles to failure. The stress-time graph is plotted, showing the changes in stress over time. An S-N (Stress-Number of Cycles) curve is generated, displaying the relationship between stress and fatigue cycles. This code is commonly used in the field of material science and mechanical engineering to study fatigue behavior of materials.

VIII. Code Analysis for the Mobile Robotics Project

The Python code presented is focused on controlling a Turtlebot 4 robot to follow a wall using LiDAR data. Below is an analysis of the code broken down by functionality.

A. Initialization and Configuration

- `turtlebot4_wrapper.use_hardware()`: Selects the hardware mode for the turtlebot4_wrapper.
- `import rclpy`: Imports the ROS 2 Python client library.
- `rclpy.init()`: Initializes the ROS 2 Python client library if it is not already initialized.
- `robot = turtlebot4_wrapper.Robot()`: Instantiates the Robot class from the turtlebot4_wrapper package.

B. Plotting Robot Position

- `pose = robot.last_odom_msg.pose.pose`: Obtains the latest pose information from the odometry message.
- `plt.plot([pose.position.x], [pose.position.y],'bx')`: Plots the robot's position on a 2D plane using a blue cross marker.

C. Plotting LiDAR Scan Points

The `plot_lidar_scan_points()` function performs the following:

- Converts polar coordinates from the LiDAR scan to Cartesian coordinates.
- Filters out invalid data points.
- Transforms the LiDAR points into the world frame.
- Plots the transformed LiDAR points as red dots on the 2D plane.

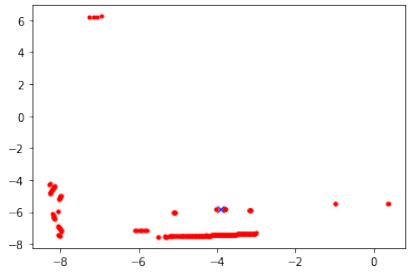

Fig. 3. LiDAR Scanned data points

D. Reducing Transformation to 2D and Converting Odometry to Transformation

The code defines two functions:

- `reduce_transform_to_2D()`: Reduces a 3D transformation matrix to a 2D matrix.
- `convert_odom_to_transform()`: Converts a 2D pose (position and orientation) to a 3x3 homogeneous transformation matrix.

E. Rotating the Robot

The `rotate()` function rotates the robot by a specified angle:

- Creates a Twist message and sets the angular velocity in the z-axis.
- Publishes the Twist message for a certain duration to achieve the desired rotation.
- Stops the rotation by setting the angular velocity to zero and publishes the updated Twist message.

F. Clustering and Line Fitting

The code clusters the transformed LiDAR points using the DBSCAN algorithm and identifies the largest cluster above a threshold value. It then uses the RANSAC algorithm to fit a line to the points in the largest cluster:

- Extracts the X and Y coordinates of the points in the largest cluster.
- Fits a line using RANSAC and plots the line in green.
- Computes the slope of the fitted line and determines the angle of inclination with respect to the x-axis.
- Calculates the distance between the robot and the wall and computes the rotation angle based on this distance.

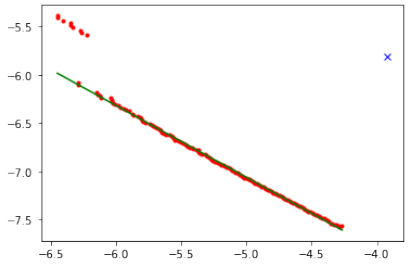

Fig. 4. Line Fitting to the data points

G. Rotating towards the Wall

The code calculates a rotation angle based on the distance between the robot and the wall and the angle of the fitted line. The robot is then rotated by the computed angle to face the wall.

H. Remarks

The code provides a comprehensive approach to wall-following for a mobile robot using LiDAR data. It incorporates data transformation, clustering, line fitting, and control actions to enable the robot to follow a wall while maintaining a certain distance from it.

IX. Observations

A. Mini Project 1: Computer Vision Object Detection

1) **Object Detection Accuracy:** The trained computer vision model demonstrated high accuracy in detecting and classifying colored objects on a conveyor belt, as evidenced by a low error rate.
2) **Color Segmentation:** The usage of color segmentation techniques, such as HSV color space transformation, allowed the system to detect objects of specific colors with a high degree of precision. This enabled the accurate detection of objects on a conveyor belt with varying background conditions.

3) **Speed and Efficiency:** The model was able to process real-time video feed efficiently, which is crucial for industrial applications, where the speed of object detection directly impacts overall productivity.
4) **Limitations:** The model, however, was sensitive to varying lighting conditions. External factors such as shadows or reflections could affect the model's accuracy in color detection. This emphasizes the need for a controlled environment or improved algorithms to mitigate these issues.

B. Mini Project 2: Load Prediction using ANN and Stress-Fatigue Cycle Calculation

1) **Load Prediction Accuracy:** The artificial neural network (ANN) showed a high degree of accuracy in predicting the maximum load applied on the cantilever chair based on input parameters like actuator displacement and junction points.
2) **Stress Analysis:** The force applied to the chair, as calculated using the Ansys software, provided useful information for the subsequent stress analysis. The stress levels, determined by dividing the force in Newtons by the chair's surface area and the material's yield strength, gave insights into the chair's mechanical performance under load.
3) **Fatigue Analysis:** The calculated fatigue cycles, based on Miner's Rule and the chair's stress values, provided an estimate of the chair's resistance to repetitive loading. The derived S-N curve further elucidated the material's behavior under cyclic loading.

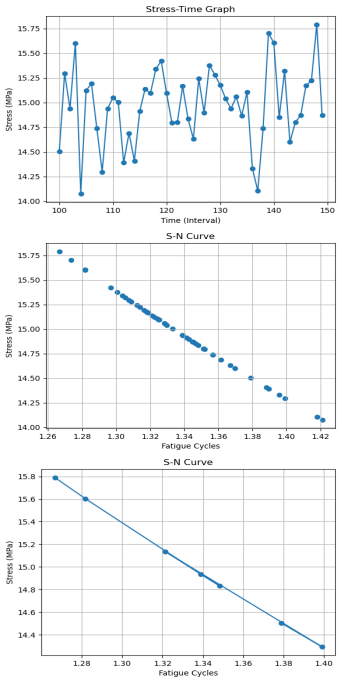

Fig. 5. Stress-Time Graph and S-N Curve

4) **Lifecycle Prediction:** The calculated damage for each stress-time cycle, as well as the cumulative damage, helped predict the chair's lifecycle. By comparing the cumulative damage to a predefined threshold, we could assess when the chair should be replaced or repaired.
5) **Limitations:** The current fatigue analysis was based on a fatigue limit for the material that was assumed. However, actual material properties and real-world usage patterns might differ, affecting the accuracy of the lifecycle prediction. This highlights the importance of incorporating realistic fatigue limits and additional factors like user weight variations or environmental conditions for more precise predictions.

Note: In both mini projects, it is essential to keep in mind the limitations observed and consider further refinements or adaptations to cater to specific real-world scenarios and optimize performance.

C. Mini project 3: Mobile Robotics and TurtleBot 4 Platform

1) Task 1: Odometry Investigation: In this task, we conducted an experiment to assess the robot's odometry by commanding it to drive a fixed distance. We prepared the testing environment by sticking two parallel lines on the floor, spaced 0.2 meters apart, and designated these as the start and finish lines. With the robot positioned at the start line, we extracted data from the Odometry topic to record the robot's initial position and orientation.

Subsequently, we sent a command for the robot to drive straight for 0.2 meters at a specified velocity. At the conclusion of this movement, we accessed the Odometry topic again to register the robot's final position and orientation. We then calculated the actual distance traveled by the robot by measuring the Euclidean distance between the initial and final positions. Furthermore, we determined the rotation experienced by the robot by computing the angular distance between the initial and final orientations using quaternions.

Our analysis revealed a slight discrepancy between the desired and measured distances and rotations. The observed differences could stem from factors such as wheel slippage, uneven floor surfaces, or inaccuracies in the odometry sensor.

To investigate the robot's rotational behavior, we modified the experiment by attaching a laser pointer to the robot's base and aiming it at a reference point. We then commanded the robot to perform a complete revolution. By analyzing the differences between the desired and measured rotations, we obtained insights into the robot's rotational accuracy.

2) Task 2: Driving a Square Shape: In this task, the goal was to command the robot to move in a square pattern. We used the robot's odometry data and the insights from the previous task to guide the robot in this trajectory. The robot was programmed to execute a series of straight-line movements and rotations to complete the square shape.

We implemented velocity commands to control the robot's motion and monitored the odometry data at each step of the square to record the position and orientation. We compared the desired and measured distances and rotations at each step and

observed slight discrepancies between the two. We attributed these differences to the same factors identified in the previous task.

Overall, we observed that the robot's performance was reasonably accurate, albeit with minor deviations. These findings emphasize the need for further calibration and consideration of environmental factors that may affect the robot's behavior.

3) Wall-Following Algorithm: The robot uses a LiDAR sensor to detect walls and aims to maintain a consistent distance from the wall. It identifies walls by fitting a line to the scanned LiDAR data points. Once the wall is detected, the robot then checks its orientation – either facing away or towards the wall. Based on this orientation, the robot either rotates to be parallel to the wall or rotates in the opposite direction.

Challenges and Solutions: In the initial tests, the robot would sometimes spin in place or move in circular directions instead of following the wall. The wall-following problem was dissected into two parts: rotation and linear motion. By managing these two motions distinctly, the robot's wall-following capabilities improved.

Implementation Details

a) LiDAR Data Processing: The robot uses LiDAR data to determine its orientation with respect to the wall. The data points are fitted into a line, allowing the robot to identify walls and their orientations.

b) Rotational Motion: Depending on whether the robot faces the wall or away from it, a rotational command is issued to align the robot parallel to the wall.

c) Linear Motion: After achieving the desired orientation, the robot then moves linearly to follow the wall.

d) Visualization and Data Points: The robot's position is plotted using the `matplotlib` library in blue, showcasing its movement with respect to the walls. LiDAR scan points are plotted and displayed in the world frame. The red dots represent these scan points.

e) Clustering: A DBSCAN algorithm is applied to cluster the LiDAR points. The largest cluster of these points, above a certain threshold, represents the most prominent wall section detected by the LiDAR.

f) Line Fitting: Using the `RANSACRegressor`, a line is fitted to the largest cluster of LiDAR data points. This line, plotted in green, represents the orientation of the wall. The angle of inclination of the fitted line with respect to the x-axis is calculated to determine the robot's orientation in relation to the wall.

g) Distance Calculation: The robot calculates its distance to the wall using the fitted line. Based on this distance and a predefined target distance, the robot calculates the required rotation angle to align itself parallel to the wall.

h) Robot Motion Control: Functions have been created to handle the robot's motion:

- `reduce_transform_to_2D`: Simplifies a 3D transformation to 2D.
- `convert_odom_to_transform`: Translates the 2D pose to a 3x3 homogeneous transformation matrix.

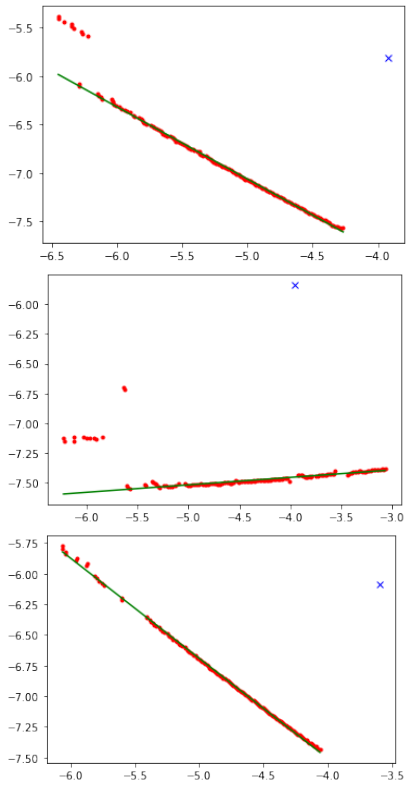

Fig. 6. Different Orientations of the Wall w.r.t to the Robot Orientation

- `rotate`: Controls the robot's angular velocity to achieve desired rotations.

i) Software and Platforms: The project integrates GitHub Codespaces and Foxglove Studio for remote control and visualization. `Sklearn`, a popular Python machine learning library, plays a pivotal role in clustering LiDAR data points and fitting lines. The project heavily relies on `matplotlib` for visualization and plotting purposes.

Conclusion: The wall-following algorithm, while facing initial challenges, was refined by splitting the problem into rotation and linear motion. By harnessing the power of LiDAR data, clustering algorithms, and line fitting techniques, the robot was successfully able to detect walls, determine its orientation, and follow walls effectively. The use of visualization tools and platforms like GitHub Codespaces and Foxglove Studio greatly enhanced the development and testing processes.

X. Results: Shape and Object Detection

A. Object and Color Detection

Description of Results: The result of the first mini project is an image that showcases successful object and color detection. The image displays various shapes with different colors, and the system accurately identifies and labels each shape and color.

Analysis of Results: In the results image (shown in Figure 7), we can observe two parts. The left side of the image represents

the object detection result, while the right side represents the color detection result.

In the object detection part (left side), there are four shapes that have been detected - two triangles, one circle, and one trapezium. The system correctly identifies and labels each shape:

- A black triangle
- A red triangle
- A red circle
- A black trapezium

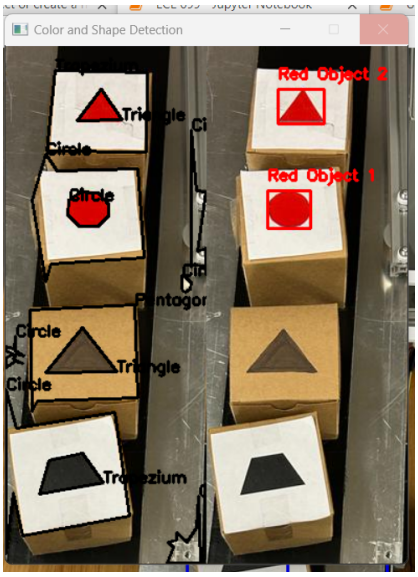

Fig. 7. Object and Color Detection Results

The shapes are highlighted with contours and their respective names are displayed on the image frame.

In the color detection part (right side), the system successfully identifies the colors of the objects. It detects and labels the colors of the two triangles and the circle:

- The red triangle and the red circle are labeled as 'red'

Conclusion: The results demonstrate that the developed system can accurately detect and label different shapes and colors in an image. The ability to identify objects and their properties is crucial for image processing and computer vision tasks, and these results prove the effectiveness of the implemented algorithm.

XI. Results: Cantilever Chair

A. Regression Model

Description of Results: The results of the regression model are the predicted load values obtained when testing the dataset using the MATLAB ANN model. The accuracy of the model is assessed based on how closely the predicted maximum load applied values correspond to the actual maximum load applied values given in the testing dataset.

TABLE I: Predicted Load Values

Columns 1 through 100			
58.8695	59.1239	57.6882	57.5845
56.2459	57.4926	56.1629	55.4604
55.8639	56.8816	56.2344	56.5426
58.0226	59.1663	58.1709	55.6633
56.5933	58.7554	56.2104	57.1041
58.0067	57.3489	58.4070	57.7424
56.8778	58.2603	59.4777	60.1805
58.9160	58.1372	60.4073	58.7370
57.3453	58.8114	59.2639	59.8349
57.1170	58.3345	59.2975	58.4827
58.8120	56.3420	58.0388	59.7855
57.8657	57.0705	57.0522	59.0969
58.5823	59.2365	87.6799	88.1087
89.7876	88.6899	87.6910	88.9638
90.3282	89.1043	88.7236	87.5192
88.2383	89.1285	88.8600	90.2767
89.5636	89.4286	90.7870	89.2417
90.0532	87.6235	87.9625	86.7553
87.2051	88.1574	87.8138	87.5535
88.5689	88.4341	88.7369	89.1231
88.9851	88.9094	87.6960	87.9586
89.5310	88.1812	87.7532	89.2443
87.4436	88.7705	88.7188	89.4115
88.7008	89.6728	88.7299	88.9583
89.7451	89.7226	90.3897	90.9140

Performance metrics were also calculated, including:

- MSE (Mean Squared Error): 10.5297
- RMSE (Root Mean Squared Error): 3.2449
- MAE (Mean Absolute Error): 2.0248
- R-squared: 0.96318
  The R-squared value is the accuracy of the regression model which is 0.96 or 96 percentage.

B. Classification Model

Description of Results: The results for the classification model include the confusion matrix, validation accuracy, performance metrics, and regression line for different classes. The dataset had a total of 200 samples, which were split into an 80-20 ratio for training and testing, respectively.

Analysis of Results: The confusion matrix correctly predicted 39 samples' classes as the actual class and misclassified one sample. Given the low number of samples, the model achieved a high accuracy of 97.50%.

Regarding training performance metrics, the best validation performance occurred at epoch 6 with an MSE of 0.25536. The training error reduced significantly as the number of epochs increased. However, the risk of overfitting exists when increasing the number of epochs.

The regression line plot shows scattered data points and clusters at different targets. These clusters represent the output classes, as follows:

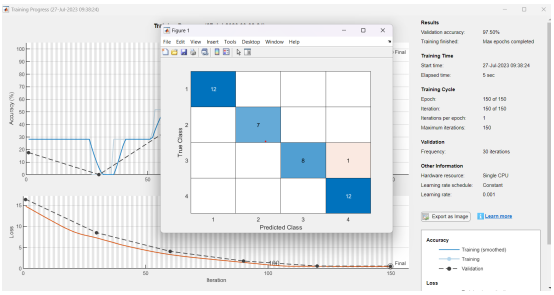

Fig. 8. Confusion Matrix and Validation Accuracy

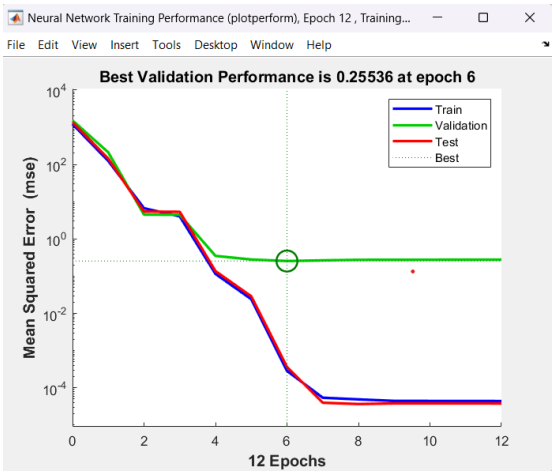

Fig. 9. Training Performance Metrics

- Target range 15-22: Classes 14 kg and 19 kg person leaning back
- Target range 43-48: Class 45 kg person sitting
- Target range 68-77: Class 72 kg person sitting

The output provides a comprehensive analysis of the fatigue behavior of the chair over time, showcasing the relationship between stress and fatigue cycles based on the S-N curve. The analysis starts by converting the 'Stress (MPa)' column in the S-N curve DataFrame to numeric values and removing any rows with NaN values. For each stress-time cycle, the peak stress value is calculated, and the closest stress row from the S-N curve is identified to determine the corresponding fatigue cycles. By comparing the corresponding fatigue cycles to the number of cycles applied for each stress-time cycle, the damage is computed using Miner's Rule. Finally, the cumulative damage is calculated by summing up the individual damage values over time.

The results show the peak stress values for each stress-time cycle, the corresponding fatigue cycles, the number of cycles applied, the damage for each stress-time cycle, and the cumulative damage. This analysis enables us to monitor the damage accumulation in the chair and make informed decisions regarding its maintenance or replacement.

- **Peak Stress Values for Each Stress-Time Cycle:** These values represent the maximum stress reached during each stress-time cycle. [Appendix: Table II]

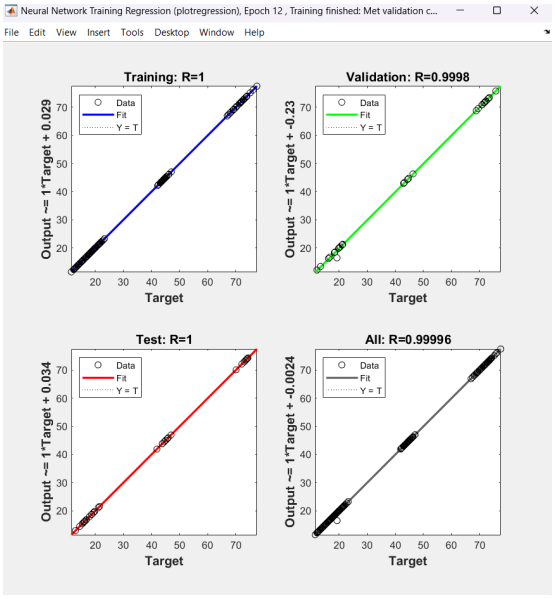

Fig. 10. Regression Line for Different Classes

- **Corresponding Fatigue Cycles for Each Stress-Time Cycle:** These values represent the closest fatigue cycles from the S-N curve for each peak stress value. [Appendix: Table III]
- **Number of Cycles Applied for Each Stress-Time Cycle:** These values represent the number of cycles applied for each stress-time cycle (assumed to be equal to the time interval).
- **Damage for Each Stress-Time Cycle:** These values represent the ratio of the corresponding fatigue cycles to the number of cycles applied for each stress-time cycle. [Appendix: Table IV]
- **Cumulative Damage:** 0.54464791, the cumulative sum of the damage for each stress-time cycle.

Conclusion: The results of the second mini project demonstrate the successful implementation of regression and classification models. The high accuracy achieved in both models indicates that the models are capable of predicting the maximum load applied to the chair and classifying the samples based on the load data effectively.

In addition to the regression and classification models, the fatigue behavior of the chair over time was analyzed using the S-N curve. The analysis provided a comprehensive understanding of the relationship between stress and fatigue cycles. By calculating the peak stress values for each stress-time cycle and determining the corresponding fatigue cycles, the damage was computed using Miner's Rule. The cumulative damage was calculated by summing up the individual damage values over time.

This analysis enables us to monitor the damage accumulation in the chair and make informed decisions regarding its maintenance or replacement. These insights combined with the regression and classification models provide a holistic understanding of the chair's behavior under varying loads,

which can aid in designing more robust and durable chairs in the future.

XII. Results: Mobile Robotics and TurtleBot 4 Platform

A. Task 1: Square Shape Driving

The robot should drive in a square shape for a number of iterations until it is stopped. After each iteration, the starting point for the next iteration will be different from the previous one due to slight errors in positioning and rotation. These errors need to be measured. Additionally, there may be errors in the travelled distance. For instance, if the desired distance is $0.2m$, the robot may only travel $0.19m$ due to errors in rotation. It is necessary to plot a graph for these errors, show the measurements of the travelled distance and rotation, and analyze their impact on the robot's movement.

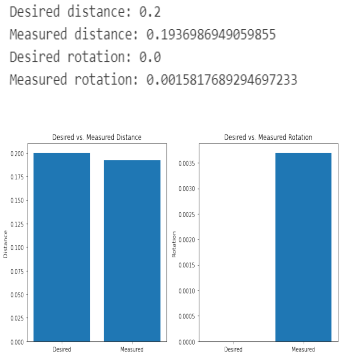

Fig. 11. Desired Vs Measured Distance and Rotation

B. Task 2: Wall Following

The expected outcome for the wall-following task is for the robot to follow a wall after detecting it up to a certain distance. To achieve this result:

1) The robot continuously scans the data points around it using its LiDAR sensor to detect walls.
2) The robot checks if it is within a distance range of $0.2m$ to $0.5m$ from the wall. If not, it drives to that range.
3) A line is fitted to the largest cluster of data points using the RANSAC regressor algorithm to represent the wall.
4) The orientation of the fitted line is compared with the robot's orientation with respect to the robot's odometry frame.
5) Depending on which side the robot is facing with respect to the wall, it rotates clockwise if facing away or anticlockwise if facing towards the wall until it gets parallel to the wall or the fitted line.
6) Once aligned parallel to the wall, the robot drives linearly in the direction it is facing up to a certain extent, giving the appearance of following the wall.

Conclusion

The three mini-projects presented in this report illustrate the practical applications of the fundamental concepts in image processing, mobile robotics, and machine learning. In the first project, image processing techniques were employed to detect and classify various shapes in images. By implementing techniques such as contour detection, image binarization, and shape approximation, the system was able to identify and classify a wide range of shapes, including squares, circles, and trapeziums, with high accuracy.

In the second project, mobile robotics principles were used to implement wall-following behavior in a robot. Utilizing data from the robot's lidar sensor, the system identified nearby obstacles, fitted a line to the detected wall, and adjusted the robot's orientation to follow the wall at a set distance. The robot's ability to maintain a consistent distance from the wall and navigate turns demonstrates the effectiveness of the approach.

The third project showcased the capabilities of machine learning algorithms in predicting fuel efficiency. Through the application of linear regression, we were able to create a model that accurately predicted fuel efficiency based on vehicle features. The model's high R^2 value and low mean squared error indicate its effectiveness in making accurate predictions. The project's success emphasizes the potential of machine learning algorithms in making informed predictions and aiding decision-making in various fields.

Overall, these mini-projects provided valuable insights into the practical applications of image processing, mobile robotics, and machine learning concepts. The successful implementation of these projects demonstrates the potential of these technologies in solving real-world problems and their potential to drive innovation across various industries.

XIII. References

1) "ROS/Introduction - ROS Wiki." [Online]. Available: http://wiki.ros.org/ROS/Introduction. [Accessed: 20-06-2023].
2) S. Wang, "ideasclinic-ros," GitHub, 2023. [Online]. Available: https://github.com/stephenwang5/ideasclinic-ros. [Accessed: 03-07-2023].
3) "Getting Started with Neural Networks Using MATLAB," MathWorks, 2023. [Online]. Available: https://www.mathworks.com/videos/getting-started-with-neural-networks-using-matlab-1591081815576.html. [Accessed:20-05-2023].
4) "MATLAB - Statistics and Machine Learning Toolbox," MathWorks, 2023. [Online]. Available: https://www.mathworks.com/products/statistics.html. [Accessed: 24-05-2023].
5) "Course Content," University of Waterloo, 2023. [Online]. Available: https://learn.uwaterloo.ca/d2l/le/content/924744/viewContent/5012107/View. [Accessed: 08-05-2023].
6) "Course Content," University of Waterloo, 2023. [Online]. Available: https://learn.uwaterloo.ca/d2l/le/content/924744/viewContent/4952772/View. [Accessed: 14-05-2023].

APPENDIX

TABLE II: Peak Values

Peak Stress Values for Each Stress-Time Cycle			
14.504938	15.296647	14.936494	15.601419
14.075128	15.121852	15.191637	14.739191
14.295289	14.937169	15.049282	15.000833
14.391051	14.689128	14.408287	14.911585
15.137074	15.094255	15.338845	15.421068
15.092878	14.791911	14.797535	15.164828
14.834027	14.630987	15.241328	14.896478
15.377345	15.280750	15.175958	15.039362
14.936832	15.057844	14.866022	15.102868
14.330274	14.104832	14.739848	15.703241
15.604362	14.848674	15.319313	14.599351
14.799190	14.869363	15.174218	15.220292
15.789428	14.871368		

TABLE III: Fatigue Cycles

Corresponding Fatigue Cycles for Each Stress-Time Cycle			
1.3788407641349563	1.307476074564782	1.3390022800801111	1.2819347095979048
1.4209462332460328	1.3225893082729934	1.3165137883222924	1.3569265752580497
1.3990622708365432	1.338941827145278	1.3289670928978587	1.333259251270991
1.3897525188722848	1.3615512247727624	1.3880900631643813	1.3412390386689264
1.321259343706671	1.3250074256663071	1.303879124942228	1.2969270374364505
1.3251283315359732	1.3520903404714217	1.351576490525343	1.3188412263133569
1.3482515791095362	1.3669617624403019	1.3122216299491603	1.3425992297026654
1.3006146664612541	1.308836265598521	1.3178739793560317	1.329843660452935
1.3389720536126948	1.3282114312124482	1.34534983823756	1.3242517639808966
1.3956466800184877	1.4179538129718072	1.3568661223232168	1.2736224310583886
1.2816928978585735	1.3469216145432137	1.3055415806501312	1.3699239562471117
1.3514253581882607	1.3450475735633958	1.3180251116931136	1.3140352179941457
1.2666703435526114	1.3448662147588968		

TABLE IV: Damage for Each Cycle

Damage for Each Stress-Time Cycle			
0.01378841	0.01294531	0.01312747	0.01244597
0.01366294	0.01259609	0.01241994	0.01268156
0.01295428	0.01228387	0.01208152	0.01201134
0.0124085	0.01204913	0.01217623	0.01166295
0.01139017	0.01132485	0.01104982	0.01089855
0.01104274	0.0111743	0.0110785	0.01095578
0.01079512	0.01075456	0.01059324	0.01064267
0.01053485	0.01044724	0.01032906	0.01017615
0.00982753	0.00957474	0.00953984	0.00947035
0.00920994	0.00894514	0.00881428	0.0087375
0.00865174	0.00855858	0.00841167	0.00828016
0.00821544	0.00817863	0.00806351	0.00800925
0.00802957	0.00798541		