Real-time Trajectory Matching of Moving Objects by an Omnidirectional Mobile Robot with YOLOv8 Object Detection Model

Ryan Barry
Electrical Engineering
Rochester Institute of Technology
Rochester, NY USA
rpb3646@rit.edu

***Abstract*—This project aims to investigate single object tracking for mobile robot navigation in real-time decision-making scenarios. A three-wheeled omnidirectional robot equipped with a deep learning computer vision architecture is used to track the trajectory of a balloon. The proposed approach involves developing a custom robot chassis, utilizing an upward-facing camera to detect and track the velocity of the balloon, and a time-of-flight sensor to measure the timing of the contact mechanism. The locomotion control system uses quadrature encoders for wheel velocity calculations, and a PID algorithm is developed to match the generated velocity vector. While the control system failed due to an issue with encoder signal edge detection, the project provides valuable insights into the design and development of a custom robot chassis for single object tracking in mobile robot navigation.**

***Index Terms*—Mobile robot navigation, single object tracking, real-time decision-making, deep learning, computer vision (CV), omnidirectional robot kinematics, PID control**

I. Problem Statement

The field of mobile robot navigation is gaining increasing attention as robots become more integrated into our daily lives. This project seeks to delve into the realm of single object tracking as it applies to mobile robot navigation, particularly in the context of responding to a moving object by making real-time decisions based on its motion. The project domain encompasses a three-wheeled omnidirectional robot equipped with a deep learning computer vision architecture that tracks the trajectory of a balloon and keeps it aloft. This application assumes operation on a flat surface, only one balloon is present at a given time, and does not consider occlusions. The proposed approach involves designing and developing a custom robot chassis using a combination of 3D-printed and aluminum components. The upward-facing camera detects and tracks the velocity of balloon, while a time-of-flight sensor is used to measure the timing of the contact mechanism. The locomotion control system incorporates encoders, and an algorithm is developed to match the generated velocity vector through PID controlled movements.

II. Literature Survey

This literature survey delves into a comprehensive exploration of diverse object detection and tracking applications, along with a consideration of prospective control systems for an omnidirectional mobile robot. Through a thorough analysis of the strengths and limitations of different detection models, this survey identifies their potential for deployment in a lightweight and time-sensitive robotics problem involving trajectory matching in real-time scenarios.

A. Noteworthy Approaches to Object Detection

Computer vision has made significant strides in developing state-of-the-art object detection and tracking models, which are becoming increasingly important in the field of robotics. As robots continue to advance and become more integrated into society, they require more sophisticated perception systems to effectively navigate their environment. Object detection and tracking models, along with other computer vision techniques, are playing a crucial role in enabling robots to interact with the world around them in a more intelligent and autonomous manner. This has led to a surge in research and development in this area over the past decade.

In the early 2000s, object detection was predominantly tackled through classical machine learning methods. However, the landscape of object detection underwent a paradigm shift in the mid-2010s with the advent of region-based convolutional neural networks (R-CNN). In particular, the researchers behind [1] introduced feature extraction on region proposals, leading to a significant 30% boost in detection accuracy compared to the existing state-of-the-art models of that time. This marked a pivotal moment in the evolution of object detection, paving the way for further improvements and innovations in the field.

Soon after the introduction of R-CNN, Spatial Pyramid Pooling Networks (SPPN) were proposed by the authors of [2]. SPPN introduced multiple pooling layers of varying sizes to enable better training performance and handle varying input image sizes. SPPN achieved similar performance to R-CNN, while offering improved computational efficiency by allowing for shared computation of features across regions.

While the R-CNN and SPPN models were important milestones in the development of computer vision, modern object detection models have far surpassed their detection capabilities. Today, the most advanced models can be grouped into

three categories: two-stage detectors, which separate object proposal from classification, and single-stage detectors, which perform both at once.

1) Two-Stage Detectors: The Faster R-CNN model, proposed by the authors of [3], is a two-stage detector that builds on the architectures of R-CNN and Fast R-CNN by introducing a region proposal network. This network extracts features that aid in detections and allows for real-time performance in various scenarios. When it was introduced, Faster R-CNN achieved state-of-the-art performance in object detection.

The authors of [4] proposed R-FCN, a two-stage detector which showed to have results similar to Faster R-CNN by proposing position-sensitive score maps to improve image classification. In their tests, their model was able to achieve similar but slightly better results to Faster R-FCN combined with the latest of ResNet, a model that excels at feature extraction [5].

[6] proposed Mask R-CNN, which adds instance segmentation masks to the existing Faster R-CNN architecture. Mask R-CNN was achieved state of the art results for instance segmentation, while two stage detectors such as Faster R-CNN remained one of the top detection models.

he authors of [7] introduced DETR, a transformer-based object detection model, in 2020. DETR adopts an encoder-decoder architecture and uses a CNN backbone to extract features, followed by a transformer and a feed-forward network for predictions. By leveraging self-attention mechanisms, DETR is able to achieve comparable results to Faster R-CNN and outperforms it on large objects. The use of transformers in object detection has shown promising results and is an active area of research in the computer vision community.

2) Single-Stage Detectors: In 2016, the Single Shot MultiBox Detector (SSD) was introduced in [8]. SSD uses a feedforward CNN to discretize the output to a set of default bounding boxes, then performs class probability estimations for each bounding box. Based on the probability scores, final bounding boxes are assigned for the predicted class. Each convolutional layer in the network predicts the bounding boxes to shape the output. In the original paper, the authors found that when the number of default bounding boxes was an order of magnitude larger than competing networks, SSD achieved a comparable state-of-the-art performance.

RetinaNet is a single-stage object detection model proposed in 2018 by the authors of [9]. It was designed to address the class imbalance issue faced by typical two-stage detectors. RetinaNet uses a Feature Pyramid Network (FPN) to extract features and generate a set of anchor boxes at each location of the feature map. The network then predicts the class probabilities and the offset values for the anchor boxes. To handle the problem of foreground-background class imbalance, the authors introduced a novel loss function called Focal Loss, which down-weights the contribution of easy examples and focuses more on hard examples during training. This is done by dynamically scaling the cross-entropy loss based on the confidence of the predicted class probabilities. RetinaNet also densely samples object locations on the input image, which is where it gets its name. Upon release, RetinaNet achieved state-of-the-art performance on the COCO dataset, solidifying its position as one of the best single-stage detectors.

You Only Look Once (YOLO) has become the gold standard for single-stage object detection in real-time applications since its introduction in [10]. YOLO and its subsequent family of networks use a single convolutional neural network (CNN) to predict bounding boxes and class probabilities. Due to its regression network architecture, YOLO can run at very high frame rates. One of YOLO's distinctive features is its tendency to predict fewer positive background bounding boxes compared to other state-of-the-art detection systems.

In early 2023, Ultralytics released YOLOv8, the latest in a family of state-of-the-art single-stage object detection networks. YOLOv8 has shown remarkable efficiency and accuracy, surpassing its predecessors in both accuracy and trainable parameters. Its improved architecture allows it to perform detections without the need for anchor boxes, resulting in more accurate generalizations. In [11], the authors compared YOLOv8 to earlier versions of YOLO using a custom dataset, and found YOLOv8 to have significantly lower training time and higher mean average precision than YOLOv7, solidifying its position as the current state-of-the-art single-stage detector. YOLOv8's speed and low complexity make it ideal for real-time applications such as this project.

B. Control of Omnidirectional Drive Robots

Mobile robotic control is a critical area of research within the robotics community. To be effective and responsive to their environment, robots require precise and reliable control of their kinematics. The following section will review some of the approaches that researchers have explored to develop such control for their robotic platforms.

[12] dives deeply into the fascinating world of omnidirectional drive robots, providing valuable insights into various approaches that designers can adopt in the development of their hardware and navigation systems. The authors have painstakingly crafted this paper with the goal of assisting in the enhancement of various aspects of the design and control systems of these highly advanced robots.

[13] introduces a novel PID control system tailored for a four-wheeled omnidirectional robot. Specifically, the control technique employed in this system is the Quantitative Feedback Theory, which allows for the design of PID speed controllers that leverage mean least squares to regulate the speed of the robot in a closed-loop configuration.

Perhaps more applicable to this current project, [14] presents an insightful account of the practical application of PID control algorithms in a three-wheeled omnidirectional drive robot, wherein each wheel operates with its own unique velocity vector. It delves deeply into the inner workings of the PID algorithm, illustrating how it harnesses the power of feedback from both IMU data and quadrature encoders to enable efficient and accurate control of the robot's motion.

Similar to [14], the researchers from [15] demonstrated the successful implementation of using IMU data to correct for

error, designing a gyroscope-based PID feedback controller to regulate the linear and angular speed of a robot. The controller is utilized to minimize the discrepancy between the desired and actual output trajectory, resulting in an optimal solution. Compared to a directly-driven PID control system, this approach yields superior performance and accuracy.

[16] investigates the performance of Linear Quadratic Tracking (LQT) and Proportional-Integral-Derivative (PID) control systems in the context of omnidirectional mobile robots and trajectory tracking. The authors demonstrate that the LQT controller outperforms the PID controller in terms of tracking a desired trajectory with higher efficiency.

The adaptive backstepping approach optimized with LQR employed in [17] is a promising control strategy for mobile robots that require fast and precise trajectory tracking. Although PID control is commonly used for overall system control, the authors show that closed-loop velocity control using adaptive backstepping can yield highly effective results. The approach is designed to achieve precise convergence of linear and angular velocity errors to zero, making it ideal for applications that require a more robust control system. While this approach may not be necessary for the slow-moving balloons in this project, it could be useful for future iterations that require faster and more precise reactions.

[18] delves into the topic of trajectory planning for omnidirectional mobile robots using linear active disturbance rejection control. It focuses on the challenges posed by the uncertainty of system dynamics of such robots when they move and rotate simultaneously. The author first presents an analysis of the robot's kinematics, and then introduces two closed loop controllers. These controllers rely on feedback from encoders and tracked trajectory to improve their performance. To address the discrepancies between the observed trajectory and the odometry-based control systems, the author employs a linear extended state observer to measure and compensate for these differences.

In this project, a PID-based controller is chosen for individual wheel velocities to match the balloon trajectory. This approach is similar to the one discussed in [13]. Although adjusting for drift using IMU data is an intriguing idea, it is deemed unnecessary because the velocity component of the balloon captured by the camera should account for the variance in the robot's path. While more advanced control systems discussed in the literature survey are interesting, a PID controller is the most suitable option for this project, as it allows the robot to remain computationally efficient.

III. Methodology

The present paper introduces a robotics project that addresses a multi-faceted problem, involving several concurrent tasks that must be completed and integrated into the final design. These tasks include training an object detection network, extracting position and velocity components of objects from detections, hardware design, and developing a control system to solve the kinematics problem required to intercept an object along its trajectory.

A. Detecting Balloons with YOLOv8

Known for its high accuracy and efficiency, YOLO (You Only Look Once) is a popular choice for real-time object detection applications. The current state-of-the-art in single-shot object detection models is YOLOv8. Upon its release in January 2023, YOLOv8 showed to be more accurate and faster than its predecessor models. This makes it perfect for a lightweight model to be deployed for single object detection in real time.

1) Data Collection and Pre-processing: The V2 Balloon Detection Dataset [19] was used to train the object recognition model in this project. This dataset comprises 75 images of balloons, each labeled with corresponding bounding box annotations. The images contain varying numbers of balloons, ranging from 1 to 28 per image. The dataset was provided in a comma-separated-values (csv) file format, with each row containing information on the image size, bounding box coordinates, number of balloons, and corresponding file name. To prepare the dataset for use with YOLO, each row of bounding box information was converted into individual text files in the YOLO format for each respective image. For training purposes, the dataset was split into training and validation sets using an 80%-20% ratio, resulting in 60 training images and 15 validation images. Additionally, a YAML file was created that included the file paths to the training and validation data, as well as the class declaration for balloons as the only class.

2) Training and Validation: In this project, the Ultralytics PyTorch training architecture was utilized with stochastic gradient descent (SGD) to train the YOLOv8 object detection model on the balloon detection dataset. The model was configured with specific hyperparameters, including an image size of 416 and a batch size of 8. A momentum of 0.85 and a weight decay of 0.0005 were also applied. Prior to processing by the network, each image was downsized to the specified image size of 416x416. This image size was chosen for its ability to achieve high frame rate detections in real-time deployment while maintaining a lightweight architecture and retaining high accuracy. The other hyperparameters were determined through several training cycles, starting from the most recent model weights, yolov8n.pt, which were previously trained on the coco dataset. The model was set to train over 250 epochs, with the training cycle ending after 236 epochs due to early stopping after not improving over 50 epochs. The hyperparameters mentioned here can be viewed below in a more structured format in Table I.

TABLE I
YOLOv8 Training Hyperparameters

Hyperparameter	Value
Image Size	416x416
Batch Size	8
Optimizer	SGD
Learning Rate	0.01
Momentum	0.85
Weight Decay	0.0005
Patience	50

3) Model Deployment and Velocity Estimation: The OpenCV AI Kit stereo depth camera (OAK-D) was used for real-time deployment of the model trained in the prior section, leveraging the DepthAI library. The OAK-D was chosen for its precise depth estimations and the ability to host machine learning pipelines on its embedded Intel Movidius Myriad X AI chip. However, since PyTorch models are not directly compatible with DepthAI or the chip, they must be converted to a compatible format. To address this, the PyTorch weights file was exported to the OpenVINO model format and then further converted to the DepthAI blob format, enabling deployment directly to the camera. This seamless conversion process ensures the compatibility of the trained model with the hardware, and enables optimal performance in real-time deployment scenarios. These factors contribute to the potential of OAK-D and DepthAI as powerful tools for a range of applications, including robotics, autonomous vehicles, and surveillance systems.

A DepthAI pipeline was developed to perform parallel balloon detections and spatial coordinate estimations. The pipeline uses the central color camera node for YOLO detections, while the left and right camera nodes function as a stereo pair for depth estimations. The detected region of interest is then used to calculate the cartesian coordinates of the detected balloons in 3D space. Additionally, the pipeline employs a weighted average method, where the 10 most recent position updates are considered with a decay rate of 20%, to compute a velocity vector in the X-Y plane, according to (1) and (2) below. The vector is weighted to favor the most recent position updates to attempt to account for the non-linearity of a balloon's velocity in real world conditions. This weighted velocity vector guides the robot to adjust its path accordingly. It is worth noting that this is the balloon's relative velocity compared to the robot. The camera frame moves with the center of the robot, so any perceived velocity is in addition to the robot's velocity at the time. Ideally, the perceived velocity is close to 0 meters per second, meaning the robot is directly under the balloon.

$$\mathbf{v}_{x,weighted} = \frac{\sum_{i=1}^{n} w_i \cdot \mathbf{v}_{xi}}{\sum_{i=1}^{n} w_i}, \quad w_i = e^{-\alpha(t_i - t_{i-1})} \tag{1}$$

$$\mathbf{v}_{y,weighted} = \frac{\sum_{i=1}^{n} w_i \cdot \mathbf{v}_{yi}}{\sum_{i=1}^{n} w_i}, \quad w_i = e^{-\alpha(t_i - t_{i-1})} \tag{2}$$

B. Hardware Design

An autonomous mobile robot was developed specifically for this project, featuring an omnidirectional drive base. This choice was made due to the unique characteristics of omnidirectional drives, which afford 3 degrees of mobility, steerability, and maneuverability. The robot was designed and constructed over the course of several weeks, with meticulous attention to detail and precision in all stages of the process. The chassis was developed through a CAD design process, which is depicted in Figure 1, and the completed robot is shown in Figure 2.

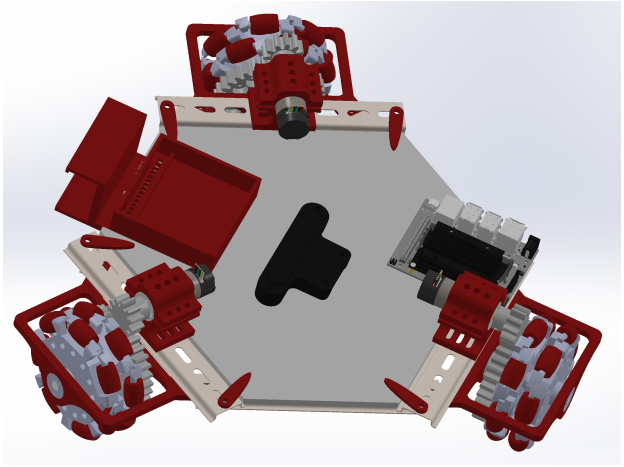

Fig. 1. Solidworks view of three-wheeled omnidirectional robot design

1) Chassis: The main chassis, constructed in a hexagonal shape, was made from eight-inch aluminum din rail. 3D printed wheel mounts were designed to securely attach to the chassis, with tight fit bearings supporting a wheel shaft. The wheel shaft was made from four-inch carriage bolts, chosen for their snug fit in 608-2RS bearings and ease of installation.

2) Omnidirectional Wheels: Custom-designed four-inch Swedish-90 wheels were 3D printed for the robot, utilizing two parts: a side panel and a roller. The side panel was crafted with seven pins arranged in a circular pattern, facilitating the support of rollers when mated in opposite directions. Each pair of panels supports seven rollers, and two assemblies are mated with a rotational offset to form a full wheel with 14 rollers, where one always touches the ground. For low-friction rotation

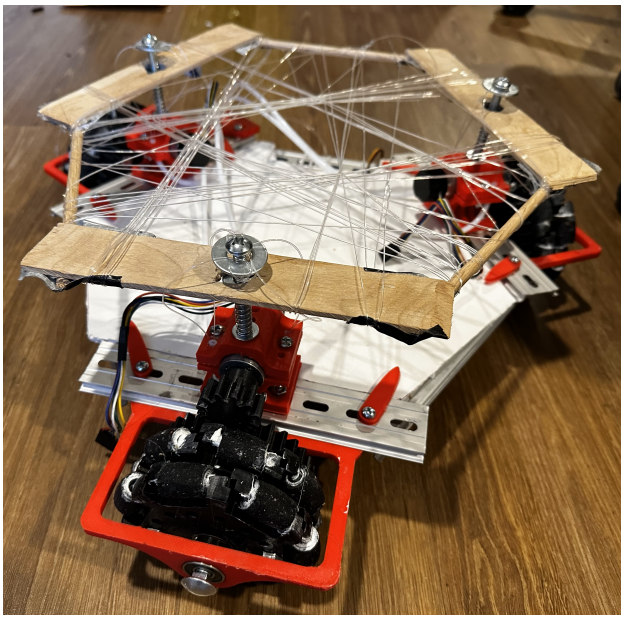

Fig. 2. Three-wheeled omnidirectional robot with spring-loaded balloon trampoline

around the axle, 608-2RS skateboard bearings fit snugly on either side of the wheel. Fig. 3 displays a partial 3D model of the wheel design. To prevent slipping caused by using bare plastic on a smooth surface, athletic tape was applied to the outside of the rollers, whereas heat shrink tubing and white lithium grease were applied to the pins of the wheel, reducing friction between the roller and the pin. Additionally, each wheel features a 33-tooth gear fastened to its side, enabling it to be powered by an 11-tooth gear on the motor shaft. Like the wheels, the gears were also 3D printed.

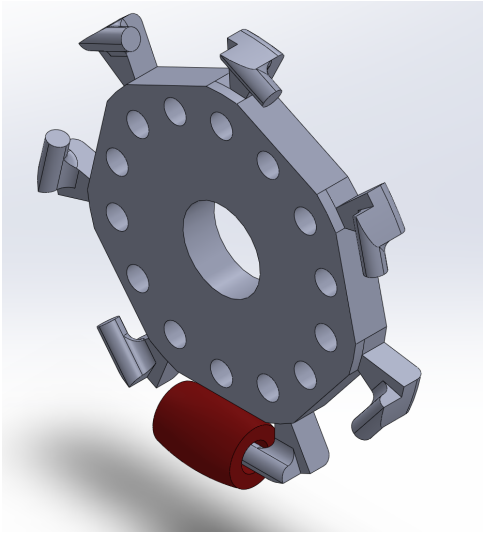

Fig. 3. SolidWorks partial view of 3D printed Swedish-90 wheel assembly

3) Balloon Launching Mechanism: The motor mounts on this robot were not only custom 3D printed parts, but also a central and versatile feature. Designed with modularity in mind, the mounts include 1/4 inch screw holes above the motor and additional holes on each corner, providing multiple mounting options for various applications. In this project, these holes were utilized to create a unique spring-loaded balloon launching mechanism that resembles a trampoline. A wooden frame, similar in shape to the robot's chassis, is wrapped with taut fishing line and supported by springs attached to each motor mount. A servo is used to compress the springs using fishing line fed through Bowden tubes. The mechanism releases the balloon once a time of flight sensor detects that it is about to reach the surface. A comprehensive view of the robot assembly, including the trampoline mechanism, can be seen in Fig. 2.

C. Power Distribution

The robot's power system utilized a 6500 mAh 2S 7.4V lithium-polymer battery, which was carefully integrated into the design. To accommodate the individual components, a custom protoboard was developed with voltage rails that support the various power requirements. The board incorporated two buck switching regulators with 5V outputs, as well as a 5V and 3.3V linear dropout regulator (LDO). With a pack rail, three 5V rails, a 3.3V rail, and a ground rail, the board provided sufficient power and voltage regulation for the system.

The Jetson Nano, which acted as the central processing unit for the robot, was powered by a single buck converter, while the OAK-D camera was powered by the other, providing a 5V output to the barrel jack. All other 5V components, such as encoders, motor controller logic supplies, and TTL logic level shifters, were powered by the 5V LDO. The use of TTL logic level shifters was required since the logic level of the Jetson Nano is 3.3V, while the encoders require a higher voltage to function properly. The low sides of the logic shifters were powered by the 3.3V rail. Finally, the DC motors that drove the wheels drew current directly from the pack rail.

D. Control System

The control system is composed of several integrated components and operates using ROS Melodic on a Jetson Nano. The ROS network includes five distinct nodes: the balloon tracker node, the forward kinematics node, the inverse kinematics node, the update target velocity node, and the wheel velocity controller node. The architecture supporting this network was designed with modularity in mind and can accommodate an N-wheeled omnidirectional robot with rollers positioned at any specified angle. A node graph graph of the ROS architecture responsible for the robot's locomotion is displayed in Fig. 4 and will be explained in the following subsections.

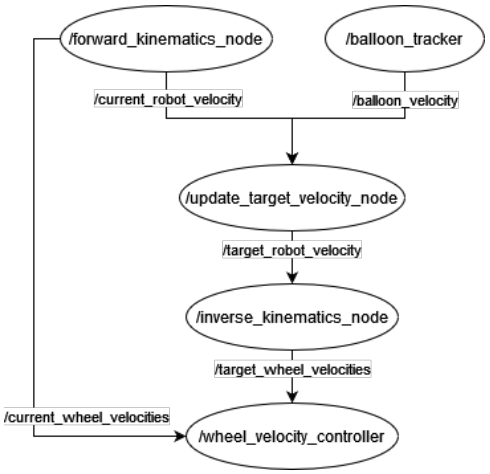

Fig. 4. Graphical view of control system ROS architecture

1) Balloon Tracker Node: The DepthAI pipeline on OAK-D employs a custom YOLOv8 model to detect balloons and calculate their velocity, as outlined in subsection A. Within the ROS network, the Balloon Tracker Node publishes the bounding boxes and velocities of the balloon to the balloon_bbox and balloon_velocity topics respectively. These topics are then utilized by other nodes to control the robot's movement.

2) Forward Kinematics Node: The forward kinematics node relies on a specialized class to compute the forward kinematics of an N-wheeled omnidirectional robot. This class leverages an instantaneous velocity function provided by a

custom quadrature encoder class to derive the robot's velocity. The quadrature encoder class calculates the linear velocity of each wheel based on the number of encoder ticks, gear ratio, and wheel radius, using Equation 1. The velocities of each wheel are published to topics corresponding to each wheel.

$$v = \text{rev/sec}_{motor} * 2\pi r * \text{gear\_ratio} \tag{1}$$

The forward kinematics class utilizes the velocities previously computed to determine the current linear and angular velocities of the robot, employing the forward kinematics equation shown in Equation 2. When the class is instantiated, the components of the equation are defined based on characteristics specific to the robot.

$$\dot{\zeta} = R(\theta)^{-1} J_{1f}^{-1} J_2 \dot{\phi}, \quad \dot{\zeta} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{2}$$

The matrix R(θ) provides information on the position and orientation of the robot in relation to the global frame. In this specific application, the global frame is always considered to be the robot's frame. Therefore, R(θ) is an identity matrix.

To derive J_{1f}, the rolling constraints for a fixed standard wheel are applied, as given by Equation 3. For a circular base with wheels evenly spaced around it, α can be calculated for each wheel (numbered 1 to n) using Equation 4. The angle β is defined as the angle between the line l from the center of the robot chassis to the wheel and the wheel's axis of rotation, and since they are colinear in this robot, β is 0. Similarly, γ is defined as the angle between the roller axis and the main wheel plane, which is 0 for Swedish-90 wheels. L represents the distance between the center of the robot and each wheel, which is constant for this application.

Using the values obtained from Equations 3 and 4, the J_{1f} matrix can be solved with Equation 5.

$$j_{1if} = \begin{bmatrix} sin(\alpha_i + \beta_i + \gamma_i) & -cos(\alpha_i + \beta_i + \gamma_i) & -Lcos(\beta_i + \gamma_i) \end{bmatrix} \tag{3}$$

$$\alpha_i = \frac{\pi}{n} + 2\pi \frac{i-1}{n} \tag{4}$$

$$J_{1f} = \begin{bmatrix} j_{11f} \\ j_{12f} \\ ... \\ j_{1nf} \end{bmatrix} \tag{5}$$

Calculating the J_2 matrix is a simple process as it involves a scaler matrix whose dimension corresponds to the number of wheels on the robot. This matrix can be obtained by multiplying an identity matrix of the same dimension by the wheel radius, which in this case is 2 inches. The vector $\dot{\phi}$ corresponds to the wheel velocities. With the help of the matrices mentioned above, the robot's linear and rotational velocities can be determined using Equation 2. Finally, these computed velocities are published to the robot_velocity topic.

3) Update Target Velocity Node: The update target velocity node utilizes the balloon_velocity and robot_velocity topics to calculate a revised velocity target for the robot. More specifically, it integrates the x and y components of the balloon's perceived velocity with the linear components of the robot's velocity to attain a net velocity of 0 m/s between the two entities. The camera frame's motion in conjunction with the robot establishes a relative velocity between the balloon and the robot, which forms the foundation of this approach's efficacy.

4) Inverse Kinematics Node: The node for inverse kinematics subscribes to the robot's targeted velocity, which is generated by the update target velocity node, and calculates the targeted velocities for each individual wheel. This is achieved through the resolution of Equation 2, which yields the velocity vector $\dot{\phi}$, thereby resulting in Equation 6. Following this computation, the velocity for each wheel is disseminated to its corresponding topic for publication.

$$\dot{\phi} = J_2^{-1} J_1 R(\theta) \dot{\zeta} \tag{6}$$

5) Wheel Velocity Controller Node: The velocity controller node for the wheels employs a Proportional-Integral-Derivative (PID) controller to achieve a match between the targeted and actual wheel velocities. The PWM signal driving the motor is updated by utilizing the output of the PID controller class. The PID control algorithm, as illustrated in Algorithm 1, is executed for every wheel. It is necessary to adjust the values of the gains, namely k_p, k_i, and k_d, for each motor to ensure that external forces are compensated without producing any oscillations.

Algorithm 1: PID Controller

Require: $k_p, k_i, k_d \geq 0$;
Initialize: $integral, derivative, error_{previous} = 0$;
Function *update*(v_{target}, $v_{current}$)
  $error \leftarrow v_{target} - v_{current}$;
  $integral \leftarrow integral + error$;
  $derivative \leftarrow error - error_{previous}$;
  $output \leftarrow$
   $k_p \times error + k_i \times integral + k_d \times derivative$;
  $error_{previous} \leftarrow error$;
end

IV. Results

A. Custom YOLOv8 Model

The YOLOv8-based balloon detection model showcased outstanding performance, achieving a remarkable final mAP50-95 score of 0.68818, surpassing even the largest standard YOLOv8 model, YOLOv8x, which scored 0.539. In contrast, the smaller YOLOv8n model, featuring 21 times fewer parameters, performed significantly worse, attaining a score of only 0.373 [20]. Additionally, the standard YOLOv8 distributions are trained on an image size of 640 compared

to this model's 416 [20]. The improved performance of the balloon detection model is attributed to its training on a single class, unlike the COCO dataset's 80 classes. This training approach is likely to have led to superior feature extraction and higher detection accuracy. However, given the small size of the dataset (75 images), overfitting may be a concern. Nevertheless, in real-world deployment, unoccluded balloons were consistently detected from the OAK-D camera's video feed without fail, indicating that overfitting did not appear to be a problem.

As depicted in the confusion matrix illustrated in Fig. 5, the model exhibits remarkable proficiency in identifying unobscured balloons, albeit occasionally struggling with misidentifying background noise as non-existent balloons. The issue was particularly evident during the deployment phase, where the model erroneously classified overhead lights as balloons, thereby impeding the efficacy of the tracking algorithm. A plausible solution to mitigate this would involve filtering out imprecise bounding boxes based on their position and recent velocity vector and minimizing system noise. Another approach could be to employ extended Kalman filtering to assess the accuracy of the detected location.

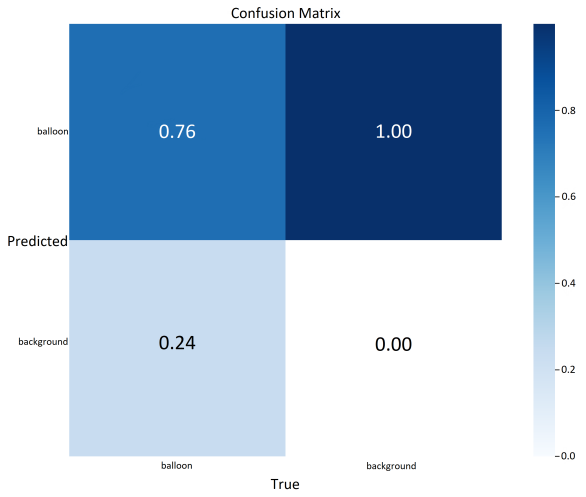

Fig. 5. Confusion Matrix for YOLOv8 model

The precision-recall curve, as depicted in Fig. 6, underscores the remarkable trade-off achieved by the model between precision and recall. At a threshold of 0.5, the curve yields an impressive mAP score of 0.821, a vital attribute for any object detection model. This achievement is a key contributor to the model's high true positive rate and low false positive rate, as shown in the confusion matrix in Fig. 5.

The F1 curve depicted in Fig. 7 exhibits an impressive equilibrium between the precision and recall of the model, at a confidence threshold of 0.511. While this threshold underwent testing, a higher value of 0.85 was discovered to be effective in eliminating most background classifications, without compromising the precision of balloon detection. Consequently, a confidence threshold of 0.85 was deployed to the robot.

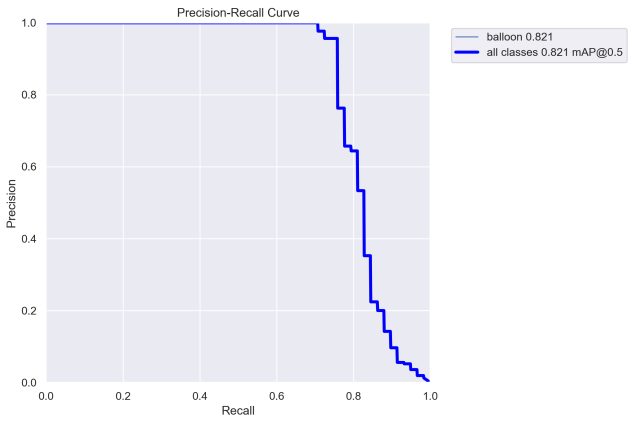

Fig. 6. Precision-Recall curve for YOLOv8 model

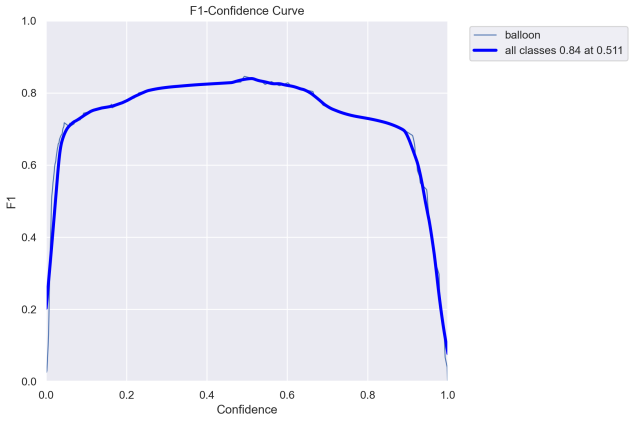

Fig. 7. F1 curve for YOLOv8 model

When deployed to OAK-D on a Jetson Nano, the model was able to detect and calculate the velocity of balloons at approximately 9 frames per second while also displaying the images. The actual processing speed is expected to be faster when the images are not being displayed.

B. Omnidirectional Drive Base

During the deployment phase, the robot encountered a challenge with detecting the pin state transitions of the encoders. This resulted in the robot's inability to measure the wheel velocities, and the velocity controller did not function as expected. In addition, the proportional, integral, and derivative constants were not tuned due to the incorrect error values caused by the absence of velocity data. Although the cause of this issue requires further investigation, it is worth noting that the robot successfully demonstrated its ability to rotate the wheels in the direction required to achieve a target velocity defined by the balloon.

In terms of hardware, the application of athletic tape around the rollers of the wheels provided sufficient friction between the wheels and the driving surface, thereby eliminating any slippage. This irradicated issues that may arise from bare

PLA not being conducive to the angling of wheels towards each other. Although fabricated through additive manufacturing techniques, the gear train from the motor to the wheel exhibited minimal backlash, resulting in a reduction of any potential systematic errors.

The efficacy of the trampoline mechanism in bouncing the balloon was found to be variable owing to the intrinsic unpredictability of the balloon's motion and orientation upon contact. Nevertheless, given that the primary objective of this project pertained to real-time trajectory matching, this issue was deemed a secondary concern and relegated to a lower priority as long as other aspects of the project were executed flawlessly.

C. Alternative Method

The proposed method for trajectory matching with a mobile robot presented in [21], utilizes adaptive color matching and Kalman filtering. The method employs tracking of the region containing the moving object by predicting the motion of pixels with the same color, and it adjusts for the effects of lighting variations. While Kalman filtering was considered for tracking in this project, it was initially deemed too computationally expensive to implement on less powerful hardware than the eventual Jetson Nano.

It is worth noting that the approach presented in this project differs from that of [21] in that the latter requires the object to be a uniform color, specifically an orange ball. In contrast, the YOLO model trained in this project was trained on images from a diverse range of colors, shapes, and lighting conditions, rendering this implementation more robust.

V. Conclusion

This project presented a novel mobile robot designed to accurately match the trajectory of a free-floating balloon. By utilizing a YOLOv8 object detection model, the robot was able to detect the balloon's position and velocity relative to itself in real-time, resulting in a trajectory vector for the robot to match. Despite facing challenges with encoder readings during deployment, the robot demonstrated its ability to effectively move the wheels in the correct direction based on the target velocity defined by the balloon.

The hardware design and construction of the robot resulted in a robust and structurally sound platform with a high degree of mobility, steerability, and maneuverability. This design has the potential to be repurposed for a wide array of applications, such as SLAM or path planning.

Future work on this project will focus on resolving the encoder reading issue, replacing the power distribution protoboard with a printed circuit board, and expanding the robot's capabilities by implementing new features such as those mentioned above.

All CAD and code for this robot has been made publicly available on GitHub [22], providing an opportunity for further development and research in the field of trajectory matching robotics. Overall, this project has demonstrated the feasibility and potential of using mobile robots for trajectory matching applications.

References

[1] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014, pp. 580-587, doi: 10.1109/CVPR.2014.81.

[2] K. He, X. Zhang, S. Ren and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 9, pp. 1904-1916, 1 Sept. 2015, doi: 10.1109/TPAMI.2015.2389824.

[3] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.

[4] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," arXiv.org, 21-Jun-2016. [Online]. Available: https://arxiv.org/abs/1605.06409. [Accessed: 06-Apr-2023].

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," arXiv.org, 10-Dec-2015. [Online]. Available: https://arxiv.org/abs/1512.03385. [Accessed: 06-Apr-2023].

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," arXiv.org, 24-Jan-2018. [Online]. Available: https://arxiv.org/abs/1703.06870. [Accessed: 06-Apr-2023].

[7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with Transformers," arXiv.org, 28-May-2020. [Online]. Available: https://arxiv.org/abs/2005.12872. [Accessed: 06-Apr-2023].

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," arXiv.org, 29-Dec-2016. [Online]. Available: https://arxiv.org/abs/1512.02325. [Accessed: 06-Apr-2023].

[9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," arXiv.org, 07-Feb-2018. [Online]. Available: https://arxiv.org/abs/1708.02002. [Accessed: 06-Apr-2023].

[10] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[11] C. Bhalerao, "Yolo V8! the real state-of-the-art?," Medium, 17-Jan-2023. [Online]. Available: https://medium.com/mlearning-ai/yolo-v8-the-real-state-of-the-art-eda6c86a1b90. [Accessed: 06-Apr-2023].

[12] H. Taheri and C. X. Zhao, "Omnidirectional mobile robots, mechanisms and navigation approaches," Mechanism and Machine Theory, vol. 153, pp. 103958, 2020.

[13] R. Comasolivas, J. Quevedo, T. Escobet, A. Escobet and J. Romera, "Low level control of an omnidirectional mobile robot," 2015 23rd Mediterranean Conference on Control and Automation (MED), Torremolinos, Spain, 2015, pp. 1160-1166, doi: 10.1109/MED.2015.7158912.

[14] T. Chaudhari, M. Jha, R. Bangal and G. Chincholkar, "Path Planning and Controlling of Omni-Directional Robot Using Cartesian Odometry and PID Algorithm," 2019 International Conference on Computing, Power and Communication Technologies (GUCON), New Delhi, India, 2019, pp. 63-68.

[15] A. Marosan and G. Constantin, "PID CONTROLLER BASED ON A GYROSCOPE SENSOR FOR AN OMNIDIRECTIONAL MOBILE PLATFORM," Proceedings in Manufacturing Systems, vol. 15, (1), pp. 27-34, 2020.

[16] A. N. Amudhan et al, "Design of controllers for omnidirectional robot based on the system identification technique for trajectory tracking," Journal of Physics. Conference Series, vol. 1240, (1), pp. 12146, 2019.

[17] G. Yang, H. -l. Liao, J. Zhou and T. Zhang, "Improved Trajectory Tracking of Autonomous Mobile Robot based on Adaptive Backstepping Control," 2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Suzhou, China, 2019, pp. 1596-1601, doi: 10.1109/CYBER46603.2019.9066496.

[18] H. Fu, L. Xin, B. Wang and Y. Wang, "Trajectory Tracking Use Linear Active Disturbance Control of The Omnidirectional Mobile Robot," 2019 IEEE International Conference on Mechatronics and Automation (ICMA), Tianjin, China, 2019, pp. 1-6, doi: 10.1109/ICMA.2019.8816299.

[19] Vbookshelf, "V2 balloon detection dataset," Kaggle, 07-Jul-2022. [Online]. Available: https://www.kaggle.com/datasets/vbookshelf/v2-balloon-detection-dataset. [Accessed: 06-Apr-2023].

[20] “Detect,” Detect - Ultralytics YOLOv8 Docs, 2023. [Online]. Available: https://docs.ultralytics.com/tasks/detect/. [Accessed: 06-May-2023].
[21] Lin Rui, Du Zhijiang, He Fujun, Kong Minxiu and Sun Lining, ”Tracking a moving object with mobile robot based on vision,” 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 2008, pp. 716-720, doi: 10.1109/IJCNN.2008.4633874.
[22] R. Barry, “Balloon Trajectory Matching Omnidirectional Robot using YOLOv8,” GitHub. [Online]. Available: https://github.com/ryan-barry-99/Trajectory-Matching-Omnidirectional-Robot. [Accessed: 07-May-2023].