Integrated Vision-Physics-Reinforcement Learning Framework for Dynamic Industrial Robot Navigation

Benyamain Yacoob, **Jingyuan Wang**, **Xinyang Zhang**

Machine Intelligence Enthusiasts

ELEE 5350: Machine Learning

University of Detroit Mercy

March 10th, 2025

Contents

1 Contributions of Team Members

Member	Contributions
Benyamain Yacoob	Discussed the design and implementation of the framework's architecture, focusing on the CNN and PINN components and their integration into the Gazebo simulation environment. Contributed to developing the RL component, particularly in conceptualizing the reward structure. Designed the architectural framework, including perception, dynamics, and decision layers.
Jingyuan Wang	Assisted in data processing by extracting and annotating video frames from the Gazebo simulation for the dataset. Prepared training materials and supported frame parsing and YOLO annotation. Contributed to developing the RL component, particularly in conceptualizing the reward structure.
Xinyang Zhang	Supported data processing efforts, including extracting and annotating video frames from the Gazebo simulation. Contributed to preparing training materials and collaborated on preprocessing tasks. Contributed to developing the RL component, particularly in conceptualizing the reward structure.

2 Accomplishments

2.1 Project Summary

This project develops an integrated framework combining vision, physics, and reinforcement learning to enable dynamic navigation for industrial robots. The work utilizes TurtleBot3 within a ROS2 Humble and Gazebo Fortress simulation environment. The framework comprises three core components:

- **Perception Layer**: A Convolutional Neural Network (CNN) for boundary detection and object recognition.
- **Dynamics Layer**: A Physics-Informed Neural Network (PINN) to model individual wheel behaviors.
- **Decision Layer**: Proximal Policy Optimization (PPO)-based Reinforcement Learning (RL) to optimize navigation strategies and wheel control actions.

This simulation-based approach facilitates thorough testing and refinement, preparing the robot to adapt to diverse scenarios. The goal is to enable the robot to detect a target object and navigate to it in a dynamic environment.

2.2 Data Processing

The data processing phase involved creating a dataset from video frames extracted from the Gazebo simulation environment using TurtleBot3. Initial simulations included obstacles, a target object, and a ground surface. Frames from the robot's camera feed were extracted and annotated with YOLOv5 framework, labeling classes such as ground, obstacles, and the target object. This produced both a raw and a labeled dataset.

We used a pre-trained model that represented a real-world dataset with approximately 80 classifications. During simulation, this model operated in real-time. After the simulation, we removed any misclassified labels because we don't want to include miscellaneous, unused labels.

We used numerous labels and classifications while exploring the simulation environment with the turtle bot. As we moved closer to objects, the classifications changed based on distance, according to our pre-trained model. Because we are analyzing video frames, we use a "super frame" technique, considering 10 frames at a time. If the majority of these frames classify an object as "human," we consider it a successful human classification. We also introduced gaps between each super frame using time series analysis. For example, we might analyze groupings of major classifications every two seconds.

When we run the simulation in Gazebo, we will select specific objects within the environment. We will keep only these objects in the dataset and then synthetically generate more images to upscale it. Our goal is to create a comprehensive dataset that focuses on the classifications we intend to use in the environment in the future.

To enhance dataset diversity, preprocessing techniques like rotations, mirroring, and brightness adjustments were applied, scaling the dataset size fourfold. This augmentation aims to improve the model's generalization across varied simulation conditions, addressing initial limitations in obstacle and lighting variability.

A camera topic was used to extract 10 frames per second from the camera's 30 frames-per-second video stream. These extracted frames were then used as raw images for the database. The database includes images of four types of modeled objects in the Gazebo indoor environment, captured from various angles, distances, and lighting conditions.

The dataset is organized into three sections: the original camera images, labels generated by the YOLO model, and the YOLO darknet file. These three sections share the same index name, simplifying later preprocessing and access.

To increase the number of samples for the CNN model, we discussed

using image flipping and adding Gaussian noise. These methods aim to expand the database more efficiently. The YOLOv5l6 results indicate that Gaussian noise does not significantly affect the results. However, image flipping causes a substantial decrease in accuracy. The YOLOv5 models used were YOLOv5x, YOLOv5n6, and YOLOv5l6. The YOLOv5l6 model was the most accurate.

2.3 Planned Implemented Methods

The framework integrates three key components tailored to the navigation task:

- **Perception Layer**: A CNN, based on a modified ResNet-18 architecture with custom boundary detection heads, processes images to identify boundaries and recognize objects, supporting multiclass classification for ground, obstacles, and targets.
- **Dynamics Layer**: A PINN, designed as a physics-constrained autoencoder, models individual wheel dynamics (e.g., turns, acceleration, deceleration), incorporating physical constraints like friction and torque into the loss function.
- **Decision Layer**: A PPO-based RL algorithm optimizes navigation decisions and wheel control actions. PPO was selected for its stability in continuous action spaces, suitable for coordinating navigation and wheel control.

The simulation environment, a custom `gym.Env`-derived setup in ROS2-Gazebo, leverages TurtleBot3 and a reliable physics engine. It features a multi-threaded sensor processing pipeline with LiDAR, an RGB-D camera, and an IMU for state estimation. The RL agent's learning will be guided by a reward structure under development, considering factors like maintaining safe distances, detecting targets, minimizing wheel jitter, and avoiding collisions. A bidirectional feedback loop integrates the components: the CNN provides boundary and object data to the RL, the PINN supplies wheel state predictions, the RL issues commands to the PINN, and the PINN supports the CNN with motion-compensated image processing.

2.4 Results and Discussion

Preliminary testing of the object detection model, trained on the augmented dataset, showed performance below pre-trained models. The model correctly identified the target object in most test cases but with lower confidence than desired, possibly due to training duration, dataset quality, or differences between Gazebo and real-world conditions. These results indicate that the

current dataset and training approach may not fully capture simulation nuances, such as lighting and texture variations.

We were using a pre-trained YOLOv5 model. We believed that switching to a more suitable model would improve the accuracy of object classification within the Gazebo simulation. Our initial tests used a real-world dataset, hoping the results would transfer to the Gazebo simulation, instead of relying on manually created images. We also considered adding artificial lighting to the Gazebo simulation to potentially enhance object classification accuracy, as poor lighting could hinder the YOLO model's performance.

We successfully parsed frames in real-time from the TurtleBot's camera sensors and wheels, even when the robot was stationary. This was achieved using our Python code, which communicates with the robot's sensors as the simulation runs. It captures real-time rotation data from the TurtleBot and combines it with camera images, sending this information to the YOLOv5 API. This API uses a pre-trained model to label the images simultaneously.

Below you can find our brainstormed framework (Figure 1), as well as some sample images (Figures 2-8) from our experiments during this initial phase of the project.

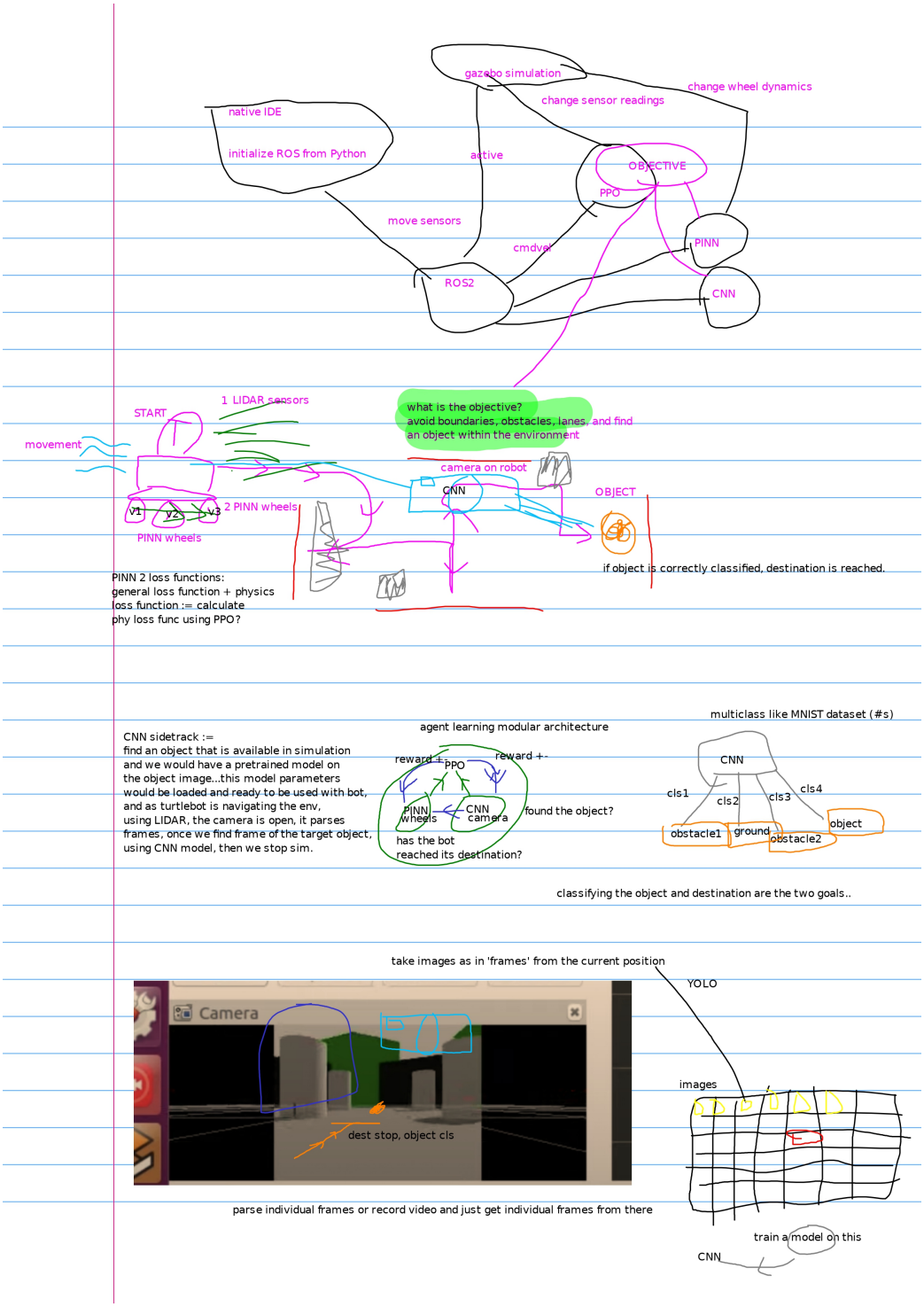

Figure 1: Brainstormed Framework

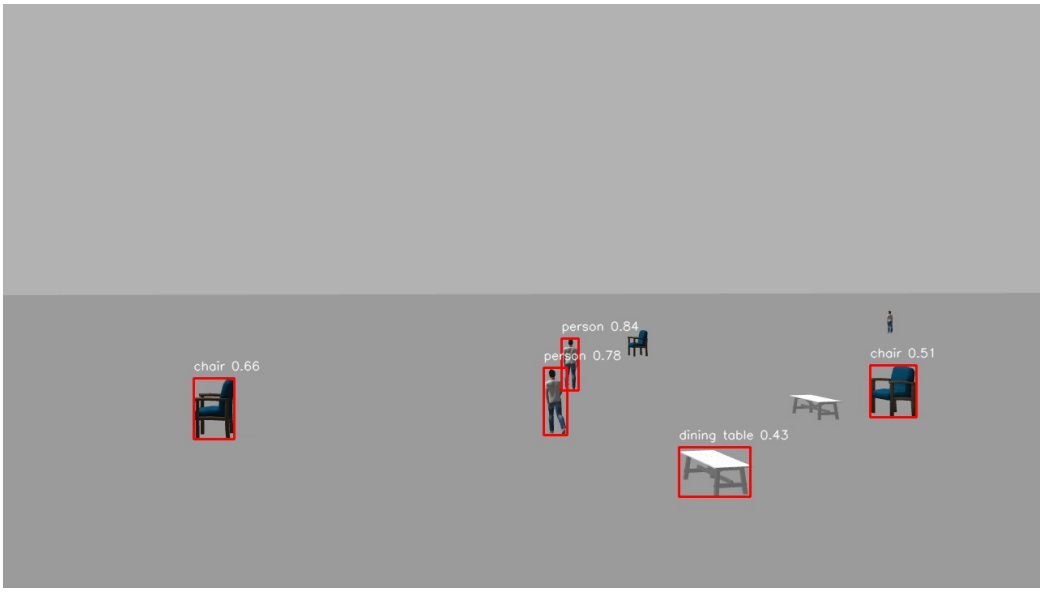

Figure 2: Object Detection Within Gazebo Sim Using YOLOv5 #1

Figure 3: Object Detection Within Gazebo Sim Using YOLOv5 #2

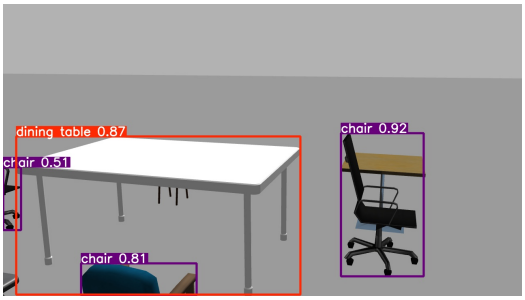

Figure 4: Object Detection Within Gazebo Sim Using YOLOv5 #3

Figure 5: Object Detection Within Gazebo Sim Using YOLOv5 #4

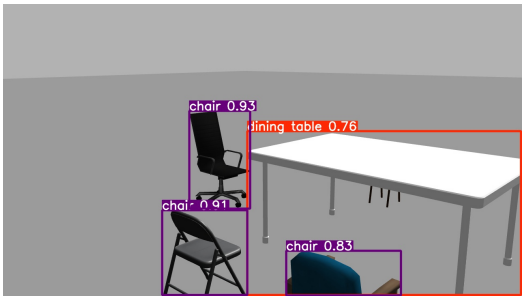

Figure 6: Object Detection Within Gazebo Sim Using YOLOv5 #5

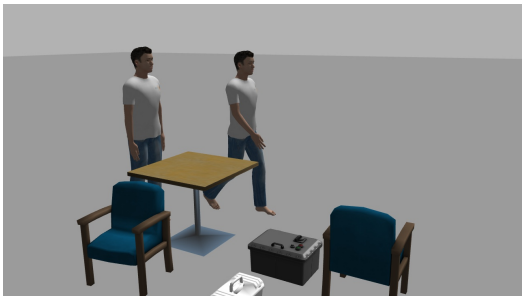

Figure 7: Object Detection Within Gazebo Sim Using YOLOv5 #6

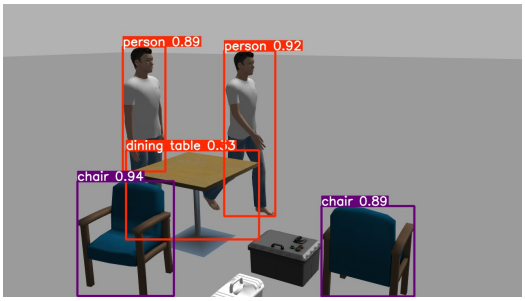

Figure 8: Object Detection Within Gazebo Sim Using YOLOv5 #7

3 Challenges Faced and Plans/Goals

A key challenge is the object detection model's suboptimal performance on simulation data compared to pre-trained models, potentially due to Gazebo's limited variability in obstacles and lighting. Additional difficulties include optimizing the training cycle and guaranteeing dataset diversity.

Another challenge we encountered is that the TurtleBot's Gazebo simulation dynamics don't match real-world behavior. This is a limitation of the bot itself, beyond our control. We should research how others have addressed this, focusing on TurtleBot configurations within the simulation. Currently, we can only control the bot's body, not the wheels directly. In a real setting, wheel dynamics are controllable. A solution could be switching to a different bot with the necessary sensors and controllable motors. We could then use it in a similar environment.

However, we want to continue using the TurtleBot and explore potential solutions, including relevant publications that use physics-informed neural networks to integrate wheel actuators on robots. If Gazebo proves unsuitable, switching to MuJoCo or Unity (as mentioned in our proposal) would be a last resort.

Figure 9 below illustrates our brainstorming process. We address the limitations of the TurtleBot simulator, specifically its constraints when applied to the real world. We also tackle the challenge of finding a suitable classification model with enough samples for our datasets and explore methods to increase the number of useful samples.

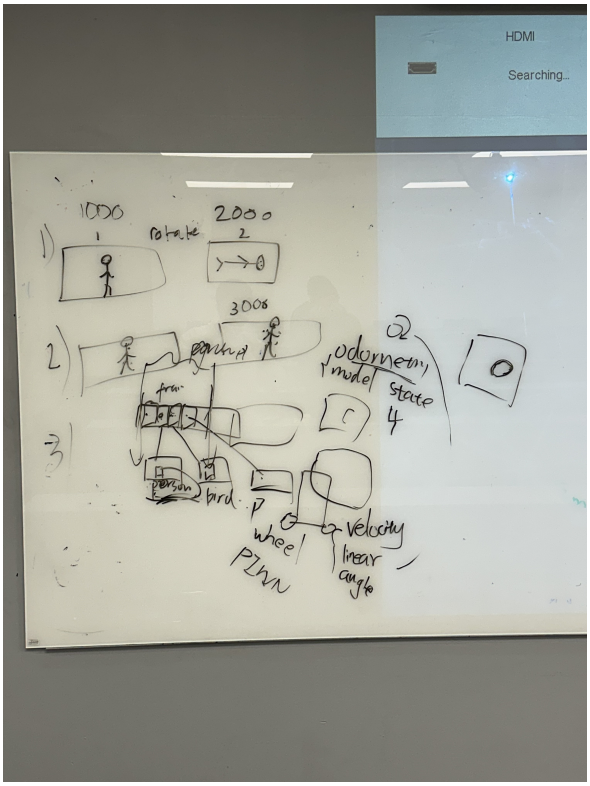

Figure 9: Current Challenges and Potential Solution

To address these, we have outlined the following plan:

- **Dataset Development**: Expand the dataset with additional Gazebo simulation frames featuring varied obstacle placements, lighting conditions, and camera angles.
- **Data Augmentation**: Apply advanced preprocessing techniques (e.g., random cropping, color jittering) to further enhance diversity.
- **Model Training**: Train the CNN from scratch on the augmented dataset over an extended period, using early stopping based on validation performance.
- **Evaluation and Refinement**: Assess performance with metrics, iteratively refining the training process.

This plan supports the project roadmap, advancing from environment setup and data collection to CNN training and eventual integration of PINN and PPO components, targeting reliable object detection and navigation.