Autonomous Navigation Using iPhone LiDAR and Deep Reinforcement Learning

EECE 5554: Robotics Sensing and Navigation

Group 4:
Kanishk Kaushal, Matthew Kazan, Sam Milhaven, and Wondmgezahu Teshome

April 20, 2025

1 Introduction

Autonomous navigation in unknown environments remains a significant challenge in robotics, often requiring expensive specialized hardware like LiDAR sensors that can have a high cost. Our project addresses this challenge by integrating two innovative approaches: utilizing iPhone LiDAR technology for environmental mapping and employing deep reinforcement learning for intelligent navigation. By combining Simultaneous Localization and Mapping (SLAM) techniques with the Information-based Distance Limited Exploration (IDLE) algorithm for global path planning and Twin Delayed Deep Deterministic Policy Gradient (TD3) for local navigation, our method provides an affordable approach that uses everyday devices to achieve effective navigation results. This integration enables robots to build detailed 3D maps of their surroundings and subsequently navigate through them autonomously, without requiring pre-existing maps.

2 Background

This project integrates two key previous works: environmental mapping using iPhone LiDAR[1] and autonomous navigation using deep reinforcement learning[2]. The mapping component provides accurate 3D representations of unknown environments, while the DRL component enables intelligent decision-making for robot navigation. Together, they create a complete solution for autonomous exploration and navigation in unknown environments.

2.1 3D Environment Mapping using iPhone LiDAR

The overall objective of this software package was to accurately reconstruct the space in three dimensions to support simulated navigation and reinforcement learning applications.

The first stage involved capturing depth data using a specialized iPhone application, which converted raw LiDAR data into individual point cloud scans. The iPhone's LiDAR scanner functions using a time-of-flight (ToF) approach, emitting infrared pulses and measuring the time it takes for the reflected signals to return after hitting an object. This allows the iPhone to estimate distances and generate accurate depth information, which is then compiled into detailed 3D point clouds. Unlike industry-standard LiDAR systems that often utilize spinning mechanical components and high-power laser systems for extensive range and precision, the iPhone's LiDAR is solid-state, compact, and optimized for shorter-range scanning suitable for mobile applications. Each point cloud thus represented a precise snapshot of the environment's geometry captured efficiently at a specific moment in time.

Next, the sequential point clouds were published and managed using Robot Operating System 2 (ROS2), which coordinated data handling, node communication, and system integration. The registration and alignment of these scans were performed using the Iterative Closest Point (ICP) algorithm from the Open3D library. Each new scan was carefully aligned with the preceding ten scans, balancing accuracy and computational efficiency. Open3D was also used to perform outlier rejection, which reduced errors caused by noise or misalignments.

To further enhance accuracy and ensure global map consistency, loop closures were addressed using GTSAM (Georgia Tech Smoothing and Mapping). GTSAM is an open-

source factor graph optimization library capable of optimizing keyframes in the global map. Once a loop closure is detected, an optimizer can be run, reworking the global map based on the new information. GTSAM significantly minimized accumulated drift, resulting in an accurate and globally coherent final map.

The integration of LiDAR scanning, ROS2-based ICP registration, and GTSAM optimization culminated in a dense, precise, and globally consistent 3D point cloud map, suitable for subsequent simulation and reinforcement learning tasks.

Finally, the completed point cloud map was saved as a PLY file, allowing easy import into Blender, an open-source 3D modeling software commonly used for creating, editing, and visualizing complex 3D models and animations. Within Blender, the point cloud data was converted into a mesh using built-in geometry processing tools. This mesh was then exported as an STL file, making it compatible with Gazebo for use as a simulation environment. This workflow is an obvious area of improvement, as there are many ways we can change it to allow a more seamless transition from data collection to practical simulation applications.

2.2 Manipulation of Mobile Robot Position using DRL

Reinforcement learning enables mobile robots to navigate by learning optimal motion policies through interaction with environments. For autonomous navigation, DRL frameworks learn to map sensor inputs to velocity commands, allowing robots to navigate toward goals while avoiding obstacles[3].

2.2.1 Robot Kinematics and DRL

The robot is modeled using the unicycle model, and the DRL model learns to manipulate the velocities (v and ω) based on the sensor inputs (e.g., laser data) and the position of obstacles. By adjusting v and ω, the robot can navigate to the goal or avoid obstacles.

2.2.2 Action Output by DRL Policy

The TD3-based actor network receives input data, which includes the robot's state (position, velocity, etc.) and waypoint data. The actor network then outputs two action parameters: linear velocity v and angular velocity ω. These actions control the robot's motion, manipulating its position as it moves towards the target.

The relationship between the wheel velocities (v_l, v_r) and the robot's motion is:

$$v = \frac{r}{2}(v_r + v_l), \quad \omega = \frac{r}{L}(v_r - v_l) \tag{1}$$

where: r is wheel radius, and L is the distance between the wheels.

Thus, the actions v and ω output by the DRL model directly control the wheel velocities, determining the robot's movement through the environment.

2.3 Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3[4] is a model-free actor-critic algorithm that improves upon DDPG for continuous action spaces by implementing three key innovations: twin critics to reduce overestimation bias, delayed policy updates, and target policy smoothing. The actor network selects actions while the twin critic networks evaluate quality by taking the minimum Q-value.

The policy gradient maximizes expected reward: $\max_\theta J(\theta) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t \mid \pi_\theta]$, with gradient $\nabla_\theta J(\theta) = \mathbb{E}_{s\sim d^\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q^\pi(s,a)]$, where $\gamma \in [0,1]$ is the discount factor and θ represents policy parameters.

3 Proposed Approach

Our method integrates 3D environmental mapping with hierarchical navigation through a two-layer architecture. The perception layer transforms iPhone LiDAR data into a 3D global map via ICP registration, while the planning layer implements a hierarchical navigation strategy combining global path planning (IDLE algorithm) with local obstacle avoidance (TD3). This separation allows our system to effectively address both the mapping challenge of unknown environments and the decision-making complexity of autonomous navigation. The system has the following components:

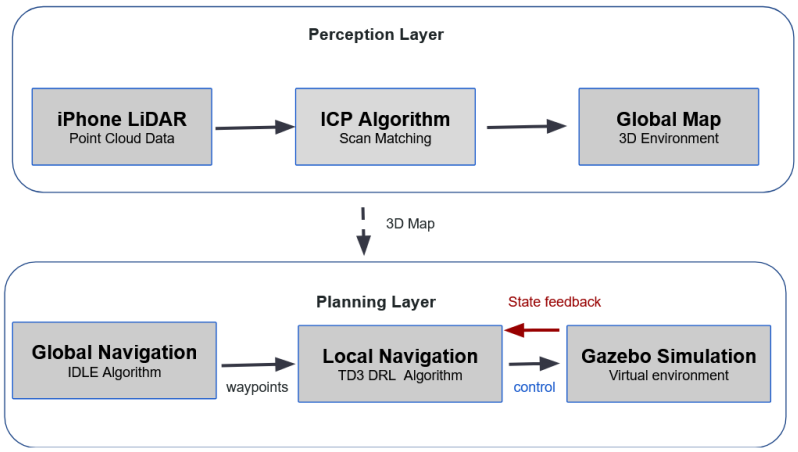


Figure 1: System architecture showing the integration of LiDAR mapping and hierarchical navigation

3.1 Global Navigation

Waypoints are selected using the *Information-based Distance Limited Exploration (IDLE)* method:

$$h(c_i) = \tanh\left(e^{\left(\frac{d(p_t,c_i)}{l_2-l_1}\right)^2} - e^{\left(\frac{l_2}{l_2-l_1}\right)^2}\right) l_2 + d(c_i, g) + e^{I_{i,t}} \tag{2}$$

Where: $d(p_t, c_i)$ is the Euclidean distance between the robot's position p_t and POI c_i, $d(c_i, g)$ is the distance from POI to the global goal g, and $e^{I_{i,t}}$ is the map information score calculated with a kernel k over candidate POIs.

3.2 Local Navigation via TD3 Implementation

The local navigation component implements TD3 for obstacle avoidance and waypoint following. Our system processes bagged laser readings and waypoint coordinates as input to the actor-critic architecture. The actor network outputs normalized action values (a_1, a_2) for linear and angular velocities, scaled according to:

$$a = \begin{bmatrix} v_{\max}\left(\frac{a_1+1}{2}\right) \\ \omega_{\max} a_2 \end{bmatrix} \tag{3}$$

The reward function incentivizes goal-reaching behavior while penalizing collisions:

$$r(s_t, a_t) = \begin{cases} r_g & \text{if } D_t < \eta_D \\ r_c & \text{if collision} \\ v - |\omega| & \text{otherwise} \end{cases} \tag{4}$$

Where $r(s_t, a_t)$ represents the reward at state s_t with action a_t, r_g is the positive reward when approaching the goal, D_t is the current distance to the goal, η_D is the distance threshold for goal proximity, r_c is the penalty for collisions, v is linear velocity, and ω is angular velocity. The $v - |\omega|$ term incentivizes efficient forward motion while penalizing excessive rotation. This local controller translates waypoints from the global navigation module into smooth, collision-free motion commands, facilitating efficient navigation toward targets while avoiding obstacles.

4 Experiment Setup

4.1 Simulation Environment

Our experiments were conducted in a 10×10m Gazebo simulation environment using custom STL world models derived from iPhone14 LiDAR scans. We implemented the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm using PyTorch within a ROS2 framework, featuring simulated 2D LIDAR. The learning agent received 21-bin laser readings combined with polar waypoint coordinates as input and produced continuous control outputs for linear (max 0.5 m/s) and angular velocity (max 1 rad/s).

4.2 Network Architecture

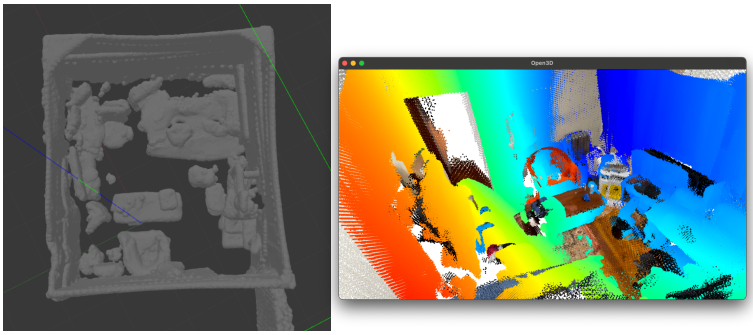
The TD3 implementation consisted of an actor-critic architecture with twin critics to reduce overestimation bias. The actor network processed 21-value laser scan vectors (180° field of view, reduced via bagging) and polar coordinates of the current waypoint through two fully connected layers with ReLU activation. The output layer produced two values with tanh activation – a_1 for linear velocity and a_2 for angular velocity – which were appropriately scaled to match the robot's maximum velocity constraints. The twin critic networks featured identical structures, each receiving state (laser scan, waypoint) and action inputs. Each critic processed states through an initial fully connected layer with ReLU activation, followed by two transformation layers for state and action features, respectively. These features were combined and passed through additional ReLU activation and an output layer that produced scalar Q-values. During training, the minimum value from both critics was selected to reduce overestimation bias during policy updates.

4.3 Training Process

Training took approximately 8 hours over 7800 episodes, with parameter updates occurring every 2 episodes to ensure stability. Episodes terminated upon goal achievement, collision events, or exceeding 500 steps. We implemented a delayed reward update window of 10 steps and incorporated Gaussian noise on sensors and actions, along with randomized obstacle placement in each episode to enhance transferability to real-world conditions.

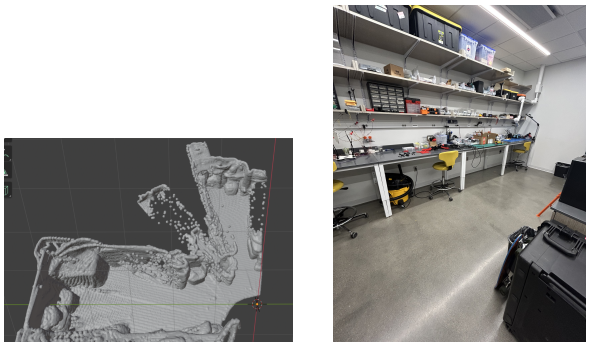
4.4 Testing Environments

We evaluated our system in three scenarios: a cluttered office environment with furniture obstacles (Fig. 2), a simple corridor with 90-degree turns (Fig. 3), and the same corridor from the opposite direction to include the glass doors (Fig. 4).



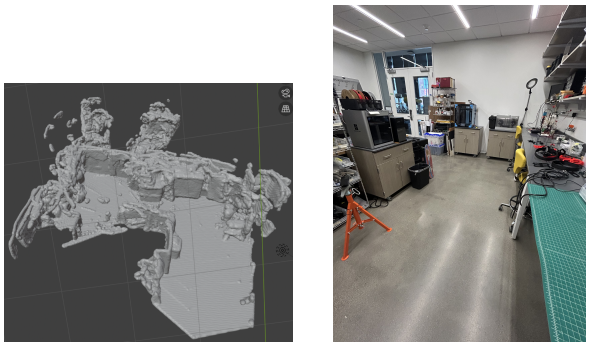
(a) (b)

Figure 2: Cluttered environment scan (a) and Rviz/PolyCam comparison (b)



(a) (b)

Figure 3: Open environment scan (a) and photo comparison (b)



(a) (b)

Figure 4: Open environment with glass scan (a) and photo comparison (b)

5 Results and Discussion

As shown in the previous figures, the iPhone LiDAR was able to scan multiple environments with a quality suitable for conducting DRL training in the Gazebo simulator. After training in each of the environments, the simulated robot was able to successfully navigate to multiple random goal positions in the environment it was trained on. These successes indicate the project's overall ability to allow users to easily take LiDAR scans of their environment and train robots to navigate within it.

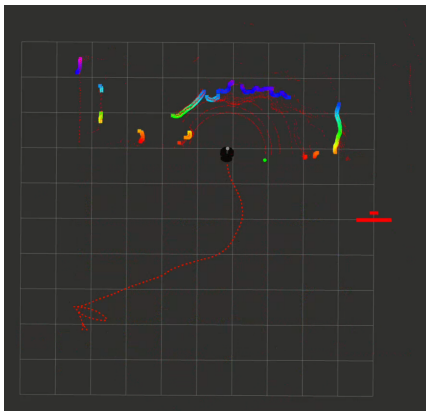


Figure 5: Example of the robot successfully navigating through the cluttered environment

5.1 Limitations

Despite promising results, our method faces several challenges. The current pipeline from data collection to simulation requires multiple manual Blender processing steps, reducing operational efficiency. Additionally, iPhone LiDAR shows performance degradation in environments with strong sunlight or non-reflective surfaces, limiting real-world applicability in certain conditions.

5.2 Future Work

Future work will focus on physical robot implementation for real-world validation, integration with advanced SLAM techniques for improved mapping accuracy, and system architecture redesign for onboard processing capabilities. Finally, develop a robust LiDAR-to-world normalization method and adapt the learned policy across multiple robot platforms to enhance generalizability.

6 Conclusion

This project successfully demonstrates a hybrid navigation system that integrates iPhone LiDAR-based mapping with deep reinforcement learning for autonomous navigation in unknown environments. By combining the IDLE algorithm for global planning with TD3 for local motion control, our approach achieves a balance of exploration efficiency and navigation reliability. Despite identified limitations, the results show promising potential for practical applications in real-world robotic navigation tasks.

References

[1] M. Kazan, *3d-lidar-slam*, `https://github.com/MatthewKazan/3D-lidar-slam`, 2024.

[2] K. Kaushal, *Drl for mobile robots*, `https://github.com/Kanishk-Kaushal/DRL-for-mobile-robots`, 2024.

[3] R. Cimurs, I. H. Suh, and J. H. Lee, "Goal-driven autonomous exploration through deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 730–737, 2021.

[4] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*, PMLR, 2018, pp. 1587–1596.