Socially Cognizant Robotic Campus Guide

Design Methods in Socially Cognizant Robotics Final Report

Aditya Kesari[1], Anis Chihoub[1], Hemali Angne[2], Jakub Suchojad[3], Sumanth Tangirala[2], Suujay Dhhoka[1], and Taqiya Ehsan[1]

[1]Department of Electrical and Computer Engineering, Rutgers University

[2]Department of Computer Science, Rutgers University

[3]Department of Psychology, Rutgers University

16 December 2024

Abstract

Navigational guidance robots face unique challenges in human-robot interaction. This project aims to enhance the functionality of a campus guide robot by improving its social navigation and digital interaction capabilities. Key improvements include real-time human motion recognition through state-of-the-art object detection (YOLOv8n) and tracking algorithms (OC-SORT), the addition of a rear-facing camera for continuous user monitoring, and an upgraded digital interface with accent-inclusive speech recognition via OpenAI's Whisper and natural language queries supported by GPT. Furthermore, navigation strategies for the robot have been optimized using search-based planning, sampling-based motion planning, and reinforcement learning techniques. A simulation environment has also been developed to enable iterative development and real-world application of socially aware navigation techniques. The paper also discusses the efficient simulation of human agent movement in the simulation environment. Overall, this work addresses the unique challenges of navigating complex, dynamic environments and aims to create an intuitive, inclusive, and adaptive robot guide. By leveraging state-of-the-art technologies and integrating multi-modal interfaces in robotics and human-robot interaction, this project contributes to existing research on socially cognizant robotics in dynamic settings.

Keywords— Human-Robot Interaction, Socially Cognizant Robotics, Digital Interface, Human Tracking, Social Navigation, Reinforcement Learning

Author names are listed alphabetically.

Email:{aditya.kesari, ac1906, h.angne, jakub.suchojad, sumanth.t, suujay.dhhoka, taqiya.ehsan}@rutgers.edu

1 Introduction

Navigating large and dynamic environments, such as university campuses, can be a daunting task for newcomers and visitors. Autonomous robots designed to guide people in these settings have the potential to provide practical solutions, given the advancements that have been made in robotics and human-robot interaction (HRI). The Socially Cognizant Campus Guide Robot is a project aimed at creating an intuitive and adaptive robot guide capable of not only assisting users with navigation but doing so in a socially aware and accessible manner.

This project is grounded in the idea that effective robotic guidance must go beyond basic navigation. It requires an understanding of human behavior, the ability to interact naturally with users, and the ability to adapt to complex environments. A socially aware robot should move intuitively in crowded spaces, predict and respond to human movements, and communicate effectively using multimodal interfaces. Addressing these aspects, our endeavor was to ensure that the Socially Cognizant Campus Guide Robot leveraged advancements in object detection, robot navigation, and natural language processing to deliver a holistic user experience.

Figure 1: Hardware setup of the Socially Cognizant Robotic Campus Guide

We use the *LoCoBot WX250S* robot platform (Figure 1) for this project. An overview of the modules we developed for the robot, along with the functions, is illustrated in Figure 2. The key technologies we used in this project include integrating advanced computer vision techniques, such as YOLOv8 for object detection and OC-SORT for object tracking, which enable the robot to recognize and follow human movements in real-time. The robot also incorporates OpenAI's Whisper module for robust speech recognition and GPT-based systems to answer natural language queries. Through these capabilities, we aimed to ensure the robot can engage with users from diverse linguistic and cultural backgrounds, making it inclusive and versatile. The development process also involved crafting social navigation methods that effectively maneuver around humans, ensuring the robot can guide users to their destinations efficiently and safely in socially dynamic environments.

The development process emphasized building a simulation environment using MuJoCo, a physics engine that allows the robot to learn navigation strategies in realistic, controlled scenarios. This simulation-driven approach enables testing and refinement of the robot and its algorithms before deployment in real-world environments, minimizing the risk of errors and improving performance. In addition, we prioritized the creation of a user-friendly interface featuring live maps, estimated arrival times, and step-by-step navigation instructions to ensure transparency and build trust with users.

The Socially Cognizant Campus Guide Robot is not just a functional tool but also a platform to explore the interplay between robotics, human behavior, and accessibility. Through its development and deployment, we hope to improve how robots interact in human environments, paving the way for broader applications in complex, human-centered environments.

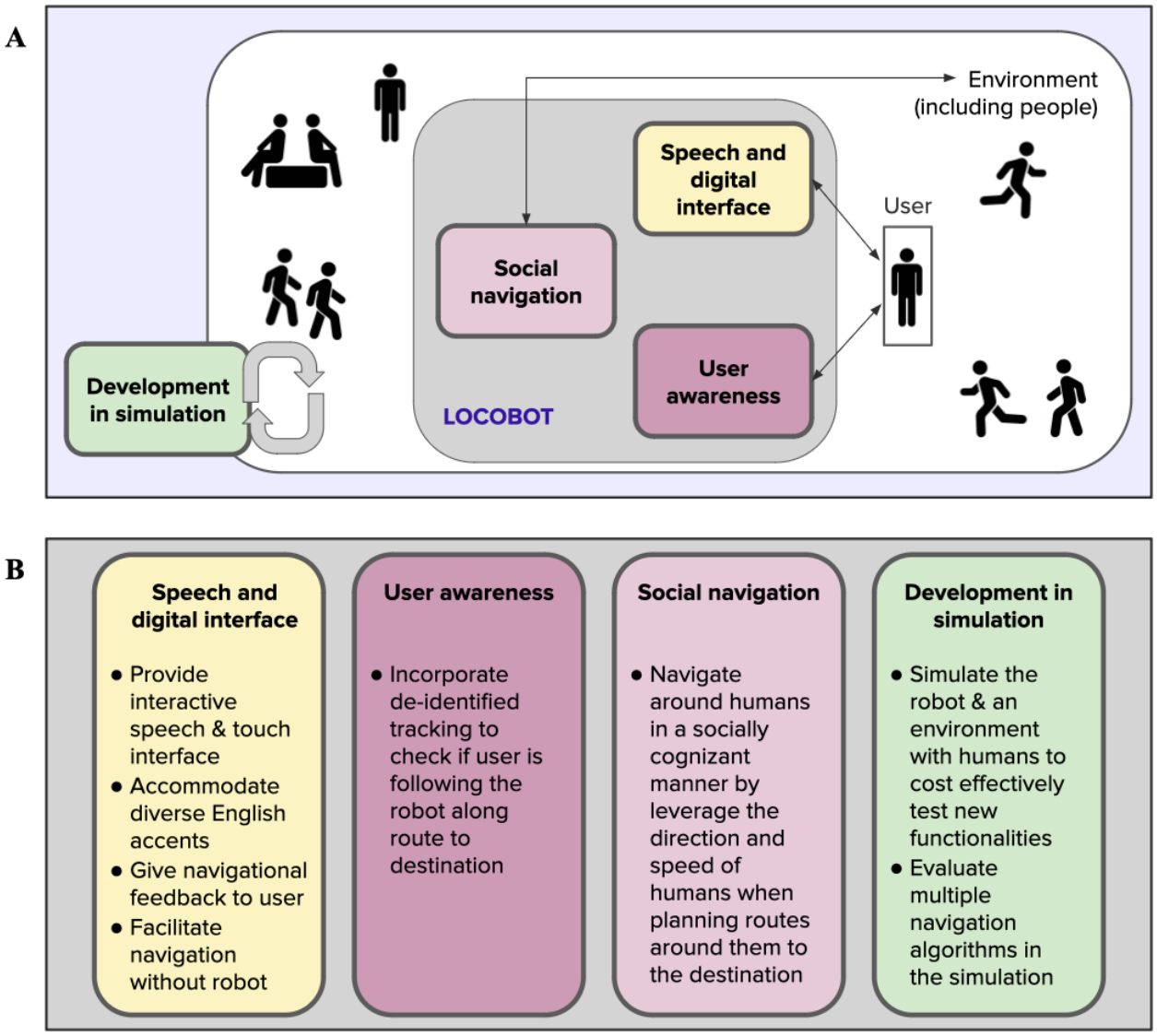


Figure 2: Overview of modules and functionalities. (A) We improved the speech and digital interface and social navigation modules of the LocoBot and added a user awareness module and an accompanying simulation environment. (B) Functionalities facilitated by these modules are briefly listed, enabling the robot to interface better with the user and other people and obstacles in its environment.

The remainder of this report is structured as follows. Section 2 presents the *Background & Motivation*, outlining the context and objectives driving the project. Section 3 discusses the *Previous Work* on the project and recaps the updates made on the robotic system by our predecessors, which we attempt to build upon through this iteration. Section 4 explores the *System Design & Modifications*, detailing both hardware and software enhancements, including subsections on *Human-Robot Interaction* (4.2.1) and *Social Navigation* (4.2.2). The section also describes the *Simulation* in subsection 4.2.3 and *ROS Integration* in 4.2.4. Section 5 focuses on *Evaluation*, analyzing the system's performance on various metrics. Section 6 highlights the *Challenges & Limitations* encountered during the project, followed by Section 7, which explores potential *Future Work* to address these challenges and expand the system's capabilities. Finally, Section 8 concludes the report by summarizing the key findings and contributions of the project.

2 Background & Motivation

Autonomous robots are increasingly being deployed in dynamic, human-centric environments to provide services such as navigation [1], delivery [2]–[4], and guidance [5]. Socially cognizant robots, designed to understand and adapt to human behaviors, represent a growing area of research driven by the demand for seamless human-robot interaction (HRI) in public and semi-public spaces [6]. These robots are particularly valuable in environments such as university campuses, where a diverse population requires intuitive and accessible solutions for navigation and engagement [7]–[9].

We introduce the Socially Cognizant Campus Guide Robot, which builds on advancements in robotics research to address key challenges in HRI, navigation, and inclusivity. Existing work highlights the importance of robots operating effectively in dynamic environments while maintaining socially appropriate behaviors [1], [10]. For example, socially aware robot navigation studies emphasize the significance of integrating human motion models to predict and adapt to crowd movement trends [11]. These approaches align with our use of advanced detection and tracking algorithms in this project to navigate crowded spaces by predicting human trajectories and adjusting their path dynamically.

Inclusivity and accessibility are central to the design of socially cognizant robots, as highlighted by prior research on adaptive speech recognition systems. For example, speech recognition systems such as OpenAI's Whisper demonstrate robust capabilities to accommodate diverse linguistic and accentual variations [12]. Studies have shown that incorporating robust speech recognition into autonomous systems improves user accessibility, particularly in multilingual environments [13]. Similarly, natural language processing frameworks, such as those powered by large language models, enable conversational interactions that enrich user engagement [14].

We also leveraged robot navigation algorithms to address social navigation challenges for this project. Advances in the field of Robot Motion Planning demonstrate the potential of these methods to enable robots to adapt to dynamic environments with minimal human intervention [15], [16]. Methods such as Rapidly Exploring Random Trees (RRT) [17] and Reinforcement Learning [18] have particularly shown promise in improving the robot's ability to independently navigate complex spaces by dynamically adjusting their paths in response to immediate environmental changes and obstacles. Simulation environments like MuJoCo [19] are crucial for developing these navigation algorithms, providing a cost-effective and safe platform to replicate real-world conditions and train data-driven navigation algorithms in complex scenarios [20], [21].

The motivation for this project is reinforced by practical needs in busy university settings. Studies on campus guide robots, such as *TritonBot*, highlight the critical importance of user engagement and trust in achieving functional and socially acceptable robotics [22]. These systems must provide clear feedback, maintain engagement during navigation, and adapt to the social dynamics of their users. Addressing these needs is pivotal for ensuring the robot becomes a valuable and trusted resource in navigating the university campus.

In this context, through Socially Cognizant Campus Guide Robot, we aim to:

1. **Enhance Social Navigation**: By using advanced detection & tracking algorithms, in combination with robust robot navigation strategies, to navigate complex environments with awareness of human presence,

enabling socially acceptable movement.

2. **Improve User Interaction and Engagement**: By leveraging advanced speech recognition and natural language processing to ensure accessibility and inclusivity.

3. **Optimize Navigation Using Simulation**: By employing simulation to prototype and refine navigation algorithms, allowing for more robust performance in real-world conditions.

4. **Foster Trust and Inclusivity**: By adapting its interaction and navigation strategies to diverse user needs, building trust through predictability and transparency.

3 Previous Work

This project builds upon the work done by our predecessors in the SOCRATES program on the *Socially Cognizant Robotic Campus Guide* [23]. The preceding work introduced a robotic system to assist students in navigating to classrooms and other locations on large college campuses.

The previous work provided a comprehensive analysis of existing implementations of robotic guides in campus and airport settings, offering a detailed review of various systems to enhance their design. It also delved into the social dynamics involved in this project, specifically the interactions between the robot and users, bystanders, and the interactions users have with bystanders in the presence of the robot. These dynamics are crucial for understanding and improving the social acceptability of robots in public spaces. Additionally, the work engaged deeply with multiple stakeholders to refine the robot's design and functionality. Broad legal considerations were also discussed, including regulatory compliance, liabilities, issues of inequity in technology use, and privacy concerns to ensure a holistic approach to the ethical deployment of autonomous guiding systems in complex environments like campuses and airports.

The hardware setup included the robotic platform equipped with a two-axis gimbal, supporting an Intel RealSense D435 camera. An Intel Nuc PC was utilized to control both the robot and the camera gimbal. Additionally, a LiDAR scanner was mounted on top of the setup for environmental scanning and navigation. A touchscreen monitor and a USB microphone/speaker acted as the user-interface components of the hardware.

The software system for the robotic guide operates on the Robot Operating System (ROS), specifically using the ROS2 Humble version. It featured a web-based user interface (UI) supported by a Python Flask backend, enabling interaction through a browser. The UI included a QR code that could direct users to an official site to locate their classes. Additionally, users could select destinations from a drop-down menu in the UI, prompting the robot to navigate to the chosen location.

Voice commands were another key feature of the previous work, implemented using PyAudio and the Google Web Speech API, allowing users to interact with the robot verbally. Speech output was facilitated through the Google Translate Text-to-Speech API. The system incorporated Fuzzy Phrase Detection to handle various queries; straightforward navigation requests prompted the robot to guide the user directly, while more complex inquiries were processed and answered using the GPT API.

To facilitate social navigation, the robotic guide incorporated person detection and navigation modules. The person detection module utilized YOLOv5, leveraging input from the camera to identify humans within the image frame and determine their positions. For navigation, the system employed the ROS Nav2 module, which could perform simultaneous localization and mapping (SLAM) using data from both LiDAR and wheel odometry. Additionally, the Nav2 module could construct a costmap that incorporated LiDAR data and dynamic obstacles, including detected humans. This costmap was continuously updated to ensure that the robot navigated around humans effectively, enhancing safety and social compliance in its operations. Additionally, various shapes of costmaps around humans were implemented to optimize the robot's movement and interaction in social settings.

The evaluation of the project revealed notable strengths and areas for improvement. The system demonstrated reliable navigation in controlled environments, accurately mapping areas and guiding users to designated locations with minimal errors. User interactions, facilitated by a web-based interface and voice commands, also received positive feedback regarding ease of use and responsiveness. However, the evaluation highlighted the challenge of ensuring seamless interaction under varied and dynamic real-world conditions, suggesting the need for further robustness in speech recognition and user interface adaptability.

The project also acknowledged several limitations. One significant challenge was the robot's ability to navigate effectively in densely populated or highly dynamic environments. The person detection module, while effective within controlled settings, demonstrated the need for enhanced accuracy and speed in real-time human detection to better facilitate interactive and autonomous navigation in busier campus environments. The inability of the module to detect the velocities of the humans also posed a challenge in implementing effective social navigation.

For the current iteration of the LoCoBot, we endeavored to build upon this existing body of work inherited from our predecessors, with the common goal of creating an autonomous, socially cognizant campus guide.

4 System Design and Modifications

4.1 Hardware

For the recent updates to our campus guide robot, we focused on adding a new Intel RealSense RGB-D camera. The camera was mounted on the back stand of the robot for stability and securely attached to prevent any unwanted movement. This addition aimed to use the previously attached camera to be repositioned to the rear, enhancing the robot's ability to monitor whether this user is still following it, maintaining its intended function of being a responsive guide. The new camera would be a replacement for the previous front-facing camera. Initially, we considered calibrating the new camera using an OpenCV script with a checkerboard pattern, but further calibration was unnecessary due to the built-in RGB-D functionality. The camera's depth sensing and integrated calibration features will allow it to accurately capture human movements without requiring additional manual calibration steps.

As part of this upgrade, we switched the orientation of the existing and new cameras. The previously used

RealSense camera now serves as the primary front-facing camera, while the new camera has been repositioned to the rear. This modification allows for improved detection and tracking of objects both at the front and rear of the robot. A new ROS-2 node was implemented to handle object detection with the newly mounted camera, which has shown improved reliability in tracking the interacting person and environmental awareness. Video recordings from both camera perspectives were captured to evaluate the overall performance, and the new configuration was found to offer visual feedback and obstacle detection in dynamic campus environments, similar to the previous camera. These videos are used to further fine-tune the code and test the YOLO module.

4.2 Software

4.2.1 Human-Robot Interaction

User Interface: While designing the current iteration of the robot's Graphical User Interface (GUI), our primary goal was to expand its base functionalities and make the interaction with the users more robust and natural. While the original version of the GUI was a simple menu for selecting the navigational destination, we imagined the robot's interface as much more. To achieve that while preserving the system's original setup of a Python-based Flask backend combined with an HTML-based frontend, we wrote additional ROS nodes and JavaScript scripts that significantly expanded the robot's functionalities. First, we introduced the *Kiosk Mode*, a GPT-powered query mechanism that allows users to ask detailed questions about campus navigation, like bus routes, estimated travel time, and detailed directions. The existing *Navigation Mode* was upgraded with new elements. The speech perception module in the *Kiosk Mode* was also incorporated into the *Navigation Mode*, allowing users to specify their destinations orally. A new "Touch Interface" consisting of two drop-down menus was also added. Users could use them to plan and select their navigational routes, specifying start and destination. Finally, we implemented an interactive visual map where users could toggle all the possible destinations as an overlay and plan routes between any two points. An overview of these functionalities is illustrated in Figure 3. Below, we describe each new addition in detail.

Kiosk Mode

Speech Perception Interface: The Speech Perception Interface (SPI) is based on OpenAI's Whisper Model [24] paired with a speaker/microphone combination device connected to the robot's PC. We wrote a standalone Whisper Node using ROS2 to handle activating the microphone, recording the speech, transcribing the recorded file, printing out the transcription to the screen, and publishing it to an appropriate topic once approved by the user. We introduced dedicated JavaScript calls in the HTML-based GUI to trigger and shut down specific processes, allowing for a controlled flow to avoid recording background noise since it remained crucial that the microphone and transcription mechanisms were active only when we explicitly communicated that fact to the user. In this early version of the SPI, the system printed the transcription out to the GUI for user approval before publishing it to the topic to which the GPT node subscribed. With enough fine-tuning of the Whisper model in the production version, we expect the performance to be robust and correct enough

Figure 3: Speech and touch interface on the robot. (A) The robot has a *Kiosk Mode* to answer general navigation questions in addition to a touch interface that provides routes to pre-defined locations. (B) The *Kiosk Mode* listens for english audio input and (C) transcribes it via Whisper. In the illustrative example shown here, the audio input was *"How to I get from College Avenue to Livingston campus?"*. (D) This transcription is then sent to a Chat-GPT Node that generates a response, often correcting for any transcription mistakes made owing to the navigational context designed for the chatbot. (E) The touch interface has pre-defined locations (F) from which a starting point and destination can be selected from a drop-down menu. (G) The map displays navigation-related information, including total distance and estimated arrival time. (H) The user also has the facility to navigate without the robot by scanning a QR code and opening the generated route on their smartphone via a mobile interface.

that the user approval step would no longer be needed, and the conversational character of interaction could be enhanced even more.

The flow of the *Kiosk Mode* SPI was as follows:

1. User taps "Ask me a question" button on the touchscreen
2. The system displays an overlay on the GUI informing the user about the SPI initialization
3. The system activates the Whisper Node, which enables the microphone
4. Once the microphone activates, the overlay message changed to "Listening to Input" for the prescribed duration (then set to 5 seconds)
5. The overlay disappears, and the system prints the transcription in the GUI, accompanied by "Approve" and "Try Again" buttons. Tapping "Approve" moves the user to the GPT module, and tapping "Try Again" moves them back to Step 1
6. Once "Approve" is tapped, the overlay is activated again, and the user receives messages about the GPT node initialization and its progress

GPT Node: The ROS2 GPT Node is based on the OpenAI's Chat-GPT[25] model 3.5-Turbo. It was equipped with a specially tailored system prompt, which contained detailed information about navigation around the Rutgers campus, allowing it to answer most of the conceivable navigation-related questions users might have. The prompt specified the possible Rutgers University bus routes and their stops, estimates of the time it takes to travel between locations, and other similar information. The prompt also prohibited the module from answering non-navigation-related questions and instructed it to politely decline if such inquiries were raised. The query resolution is carried out via OpenAI's API mechanism, which allows for seamless processing and presentation of the answer back on the screen.

The current flow of the GPT Node is as follows:

1. Once the user hits "Approve" in step 6 of the SPI Flow, the overlay and the GPT node are activated
2. The GPT Node is subscribed to the "Transcription" topic, to which the SPI published the contents of the Whisper transcription
3. The text gets sent up to Chat GPT for resolution, and the received response is printed out in the newly activated GUI space for the user to read

The future iteration should provide an option where the answer is also read aloud for the user, allowing for a more conversational flow of the interaction. The system prompt should also be updated periodically based on user input.

Navigation Mode

Speech Perception Interface: The *Navigation Mode* SPI utilizes the same Whisper Node as the *Kiosk Mode* but uses it for different purposes. The *Navigation Mode* at large is designed to allow the robot to guide the user to the selected destination. The SPI allows the user to say the location name out loud, and the robot should recognize it properly. The SPI is equipped with a case-insensitive "fuzzy matching" mechanism, providing users some leeway in specifying their destination. For example, "Busch Student Center" should be matched for entries like "BSC" or just "Student Center". Depending on its deployment environment, these fuzzy matching criteria will have to be tailored to each robot. The SPI is also paired with the map element, and once a location is matched, it will be plotted on the map.

The current flow of the Navigation SPI is as follows:

1. User taps the "Where do you want to go?" button on the touchscreen

2-4. Same as in *Kiosk Mode* SPI

5. Transcription is matched to the Location List
6. If the match is unsuccessful, feedback is given. If the match is successful, the GUI displays "Do you want to go to ⟨location⟩?" with "Yes" and "No" buttons
7. Tapping the "Yes" button triggers the "send location" function, which passes the "goal pose" (destination) coordinates to the robot and begins navigation. Tapping "No" restarts the procedure from Step 1

Touch Interface: The Touch Interface is an alternative, more robust way of using the robot's navigational capabilities. It was designed to accommodate people who prefer to interact with the robot more traditionally, and also those who are unsure about their preferred path. The Touch Interface consists of two drop-down menus, one where the user selects their starting point and one where they select their destination. The starting point will default to the robot's current location. Still, since the Touch Interface allows for path visualization on the map, users can pick any two destinations from the robot's memory and visualize a path between the two selected locations. After the route is planned and plotted on the Map Element (see details in the section below), a new button appears on the screen, allowing the user to commence navigation if they are happy with the planned route. If not, the users can simply select different destinations and re-plan the route.

The current flow of the Touch Interface is as follows:

1. User selects their preferred start and stop destinations using two drop-down menus
2. User taps "Plan Route"
3. The linear path between the two spots is calculated and visualized
4. User is asked whether they would like to start navigating to the selected destination. Tapping "Yes" begins navigation and sends the appropriate goal pose to the robot

Map Element

Main Leaflet Window: The Map Element is the central piece of the refreshed *Navigation Mode* setup. It allows for the visualization of campus locations as well as projecting routes between them, accompanied by boxes with general information about the route and detailed navigational steps. The Map Element was implemented via a JavaScript open-source library Leaflet, combined with the Open Street Map (OSM)[26] and the Open Source Routing Machine (OSRM)[27]. OSM provided a detailed and dynamic map of the environment, and the OSRM allowed for the planning of GPS-based routes between the selected locations. All three libraries work seamlessly with each other, making the route-planning process straightforward to grasp. The Map Element has three functional buttons overlayed on top of it. The top button toggles the visualization of all programmed locations as points on the map, the middle button clears the planned route and returns the interface to the default, while the bottom button generates a QR code that allows the users to send the planned route to their smartphones. The routes are visualized as dashed lines between the two selected locations. They tend to follow paved and easily traversable routes, making them robot-friendly if the user decides to have the robot take them to their destination.

Info Box: The Info Box is plotted on top of the Map Element and provides rudimentary information about the planned path obtained from the OSRM along with the route. The Info Box contains the following information:

1. Distance (feet + miles)
2. Estimated time if the user walks on their own

3. Estimated time if the robot guides the user

4. Expected number of turns

Details contained in the Info Box are important for users who make choices about their route and navigation mode. If they are unfamiliar with the campus and have enough time, they might decide to use the robot's navigational feature. But it is conceivable that they are in a hurry and the robot-based navigation might be too slow – thanks to the Info Box and the provided estimates, the users can make an informed decision about which option would work best for them instead of abandoning the robot midway through the navigation process if it turns out it is taking longer than they expected.

Navigational Steps: Navigational Steps is another box plotted on top of the map, providing additional information about each segment of the currently visualized route. Users can tap on each segment in the box, and the map will zoom in on that particular fragment and plot a thick blue line on top of the map (as opposed to the standard dashed line used for route visualization). Additionally, the box provides the distance (in feet) of each particular route segment, allowing the users to explore their expected path in more detail as well as identify some important landmarks along specific segments in case the users are not familiar with the particular campus.

QR Code Generator and Mobile Interface: The built-in QR Code Generator has been implemented to further expand the robot's capabilities by allowing the users to transfer the planned routes from the robot onto their smartphones, in case they need help navigating towards their destination but do not wish the robot to guide them there physically. Once the desired route is planned via the robot's interface, the users can tap an on-screen button with a smartphone icon, and a QR code will be generated and presented on the screen. The QR code leads to a website where a specially customized mobile interface will present the route for which the code was generated. The mobile interface consists only of the map element, enriched with geolocation and the functionality to help users navigate and follow the route. Removing the Touch Interface and the *Kiosk Mode* from the Mobile Interface enhanced the readability and usability by making the screen less cluttered while using geolocation, which improves the chances of successful navigation.

Visual Awareness

Tracking Interacting User To enhance the robot's visual awareness and support effective user guidance, we designed a robust multi-person tracking pipeline capable of maintaining real-time tracking and interaction awareness. Once the user selects their desired destination, the robot plans a route and executes it. While the system is envisioned to use real-time tracking to ensure the user is following along as it navigates – mimicking a human guide – this functionality was not fully implemented due to time constraints. The proposed design was based on principles detailed in Section 4.2.2 and would utilized the unique tracking ID assigned to the user by the tracker to maintain tracking continuity. Our approach integrated real-time detection with

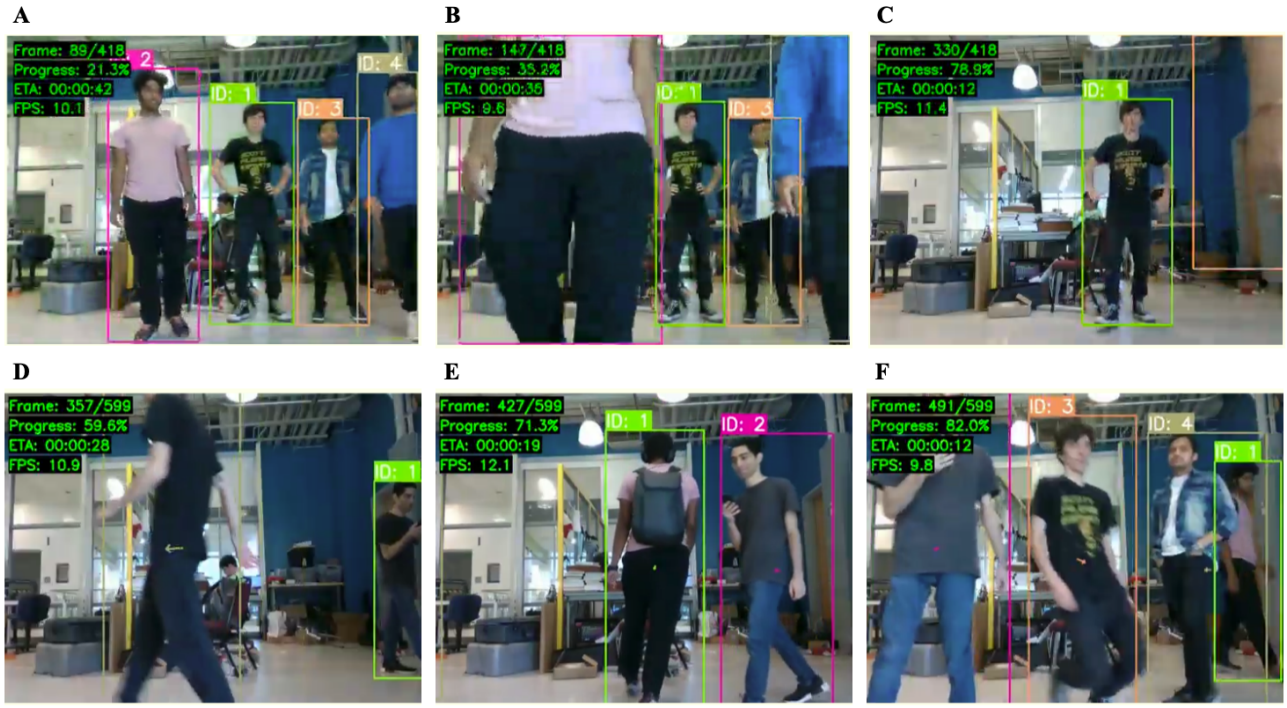


Figure 4: Person tracking and inferring direction of movement with YOLOv8 and OC-SORT. (A) A unique track ID was assigned to every person detected with confidence above the threshold. (B) The track ID remained consistent despite occlusion. (C) The ID persisted throughout the video stream. (D-F) The positions of the same ID across frames were used to estimate the direction of movement, represented by arrows. The speed was estimated as the magnitude of change in the inferred direction.

adaptive tracking using the Online Continuous SORT (OC-SORT) algorithm [28], optimized for dynamic motion scenarios.

1. **Detection and Feature Extraction:** Person detection was performed using YOLOv8 [29], configured with a confidence threshold of 0.8 to ensure high precision. Upon detection, each individual's appearance was characterized using robust feature descriptors:

   (a) **Color Features:** Processed in HSV color space with normalized 8x8 hue and saturation histograms, providing resilience to variations in lighting conditions.

   (b) **HOG Features:** Histogram of Oriented Gradients (HOG) applied to 64x128 windows captures the pose and shape of individuals through edge orientation distributions, enhancing movement understanding and re-identification.

   These feature descriptors ensured reliable tracking and consistent person re-identification across frames.

2. **Tracking with Adaptive Association and Motion Consistency:** Continuous state estimation for each detected individual was achieved using an enhanced Kalman filter [30], implemented with a seven-dimensional state vector (x, y, width, height, velocity along x, velocity along y, velocity for scale change) and a four-dimensional measurement vector (x, y, width, height) to account for different motion dynamics. The tracking framework integrated a multi-metric association process:

(a) **Spatial Association:** IoU-based position weighting with adaptive thresholds that are dynamically adjusted based on movement intensity.

(b) **Appearance Matching:** Feature similarity was evaluated through historical frame analysis, focusing on recent pose changes to sustain identity consistency.

(c) **Motion Consistency:** Velocity prediction coupled with pattern recognition ensured continuity despite rapid posture changes during dynamic movements.

3. **Dynamic Motion-Specific Adaptations:** To accommodate the unique demands of a college campus scenario, the system dynamically modified association thresholds, prioritized appearance features during high-motion states, and adjusted predictions to account for vertical and rapid movements. These adaptations improved robustness and tracking accuracy under dynamic conditions.

4.2.2 Social Navigation

Human Tracking and Direction Estimation We performed obstacle detection during navigation using a LiDAR scanner. To ensure the path planner treats humans distinctly from other obstacles, a specialized object detection module was required to identify humans within the environment. This module employed the pre-trained YOLO (You Only Look Once) model [31], designed for real-time object detection and classification. YOLO leverages a sequence of convolutional blocks, where kernels compute dot products over sliding image patches to extract features. During initial model pre-training, these kernel parameters were optimized to enhance feature extraction and object classification accuracy.

The YOLOv8n model architecture consists of three primary components: the backbone, neck, and head. The backbone includes convolutional layers alongside two additional block types that (*i*) merge features across layers to enhance detection and (*ii*) capture multi-scale spatial information for objects of varying sizes. The neck integrates these processed features, while the head generates bounding boxes and classifies detected objects. This architecture enables separate detection for small, medium, and large objects, allowing YOLOv8n to handle a wide range of object sizes within the same image [29].

Following object detection, we implemented object tracking to determine the direction of movement of each detected human across video frames. We utilized OC-SORT (Occlusion-aware Simple Online and Real-time Tracking) [28], which integrates a Kalman filter [30] to predict an object's future position based on its previous states. This approach ensured robust tracking even under temporary occlusions. In OC-SORT, detected objects are matched to predicted positions by comparing bounding boxes and appearance-based HOG (Histogram of Oriented Gradients) features, which describe object shape and appearance based on edge orientation distributions. Unlike other implementations that employ intermediate convolutional features, we exclusively used HOG features to expedite tracking.

The estimated direction of movement of each tracked human is derived by analyzing positional changes across consecutive frames. This directionality information informed the cost-map adjustments, introducing high-penalty regions in the predicted direction of movement of humans to facilitate socially aware navigation. The speed was approximated as the magnitude of the change between consecutive frames. Figures 4D–4F

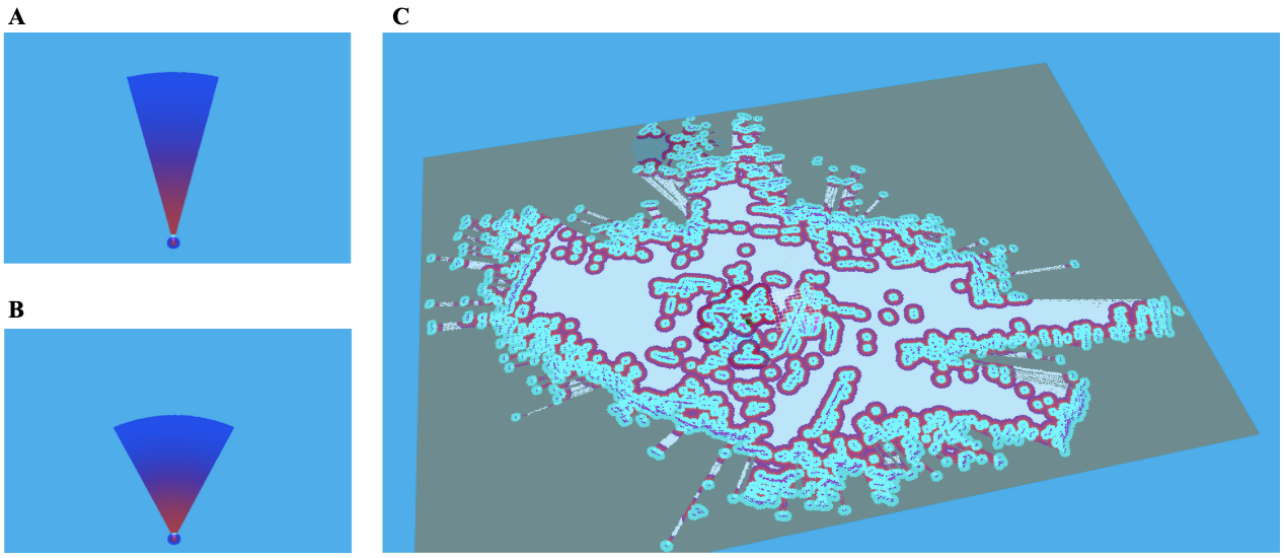


Figure 5: Cost region around detected humans and a cost-map of the environment. We replicated the design of the pizza-slice cost regions centered around the detected humans from Greenberg et al.[23], with the addition of utilizing inferred direction and speed of humans in the scene to decide the shape of the slice. (A) A faster speed gives a slice with a small angle and a large radius to reflect greater certainty of movement in that direction. (B) A slower speed renders a larger angle and smaller radius to accommodate any change in direction the human might take. A higher penalty, in red shading, is assigned closer to the human, and it gradually reduces along the radius. (C) The LiDAR scan of the environment obtained from the LocoBot is updated with the cost regions generated around the detected humans in the scene.

illustrate the tracking of humans across video frames captured by the robot's camera. Arrows indicate the estimated direction of movement of each individual.

Updated cost-map: The implemented cost region for obstacle-humans is elaborated on in Greenberg et al.[23]. We used their 'pizza-slice' cost region and inferred its shape using the direction and speed given by the updated object detection and tracking algorithms. Specifically, a human's inferred direction of movement determined the direction in which the pizza slice faces, centered around the detected human. The cost region's angle is inversely proportional to the inferred speed, while the radius is proportional to it. A faster speed implies a larger radius in which the human could be present in a given time and a smaller angle of deviation due to greater certainty of motion in that direction (Figure 5A). If the human is going slower, the angle of the "pizza-slice" is larger to anticipate any turns that they might take owing to their slower speed; the radius is smaller because they cover a smaller distance in a given time frame(Figure 5B). A penalty assigned inside a cost region gradually reduced as the distance from the center increased(red shading signifies a higher penalty closer to the center in Figures 5A–5B). The cost regions for all the humans detected in the frame were added to the cost-map obtained from a LiDAR scan of the environment(Figure 5C) when the LocoBot navigated. This combined cost-map was used to plan a path in the environment to the destination using the A* algorithm described in the next section.

Robot Navigation: This section delves into the technological frameworks and methodologies developed to enable autonomous robot navigation. We explored various strategies to ensure efficient and safe navigation in dynamic environments, particularly when interacting with human elements.

Problem Setting: The navigation challenge here was predicated on the assumption that a user, through a digital interface, has set a specific goal position, which could be a particular location on the campus. The robot should then guide the user to the target location as efficiently and safely as possible. The robot navigation algorithm must utilize the coordinates from the GPS and SLAM module and the locations of humans in the environment provided by the human tracking module to navigate and guide the user to the goal position. It must do so by dynamically processing real-time data to avoid collisions with humans, effectively balancing speed and safety in a fluid environment where human movement introduces variable obstacles.

Search-based Navigation - A*: We used the LiDAR scan to discretize the environment into a grid, transforming it into a graph where each cell is a node linked to its neighbors. This graph setup allowed for the application of graph-search algorithms, like the A* search from Greenberg et al.[23] and first proposed by Hart in [32], utilizing distance to the goal as a heuristic. The updated cost-map was integrated into these searches to help avoid both human and environmental obstacles while navigating.

This graph-search approach extended to simulating human agents as well, as outlined in Section 4.2.3. Applying it to both robots and humans turns the problem into a multi-agent navigation scenario, notorious for its high computational demands due to the need to compute multiple paths simultaneously.

To enhance efficiency, we pre-computed cost-maps for fixed goal locations of all agents, focusing solely on static obstacles. These pre-computed maps were stored and later, during online operations, updated with dynamic data relevant to each detected agent's current state. This localized update strategy significantly cut down on computation, allowing for real-time path adjustments without continually regenerating entire cost-maps. In the simulation environment (described in Section 4.2.3), generating a cost-map for each goal position over a 20 sq.m area, with a cell granularity of 0.0001 sq.m, required approximately 195 seconds on a MacBook Pro with an M1 Pro chip and 16 GB of RAM. Each cost-map file occupied 8.5 MB of storage. This approach proves more computationally efficient for multi-agent navigation.

Limitations of Discrete Path Planning in Holonomic Robots: While the algorithm described above is feasible for navigating the robot, it has notable limitations that result in sub-optimal path planning, especially for a holonomic robot. A holonomic robot can freely control all of its degrees of freedom [33]. In the case of a holonomic locomotion robot, like the LocoBot, the robot can independently alter any of its three degrees of freedom — the X and Y positions and its orientation — without needing to adjust the others. This means it can rotate in place, unlike a car (a non-holonomic system), which must generally be in motion to change direction. However, holonomic robots still require an initial rotation to align with the desired direction before moving toward a target.

1. **Movement characteristics and alignment:** The characteristic of holonomic robots is not optimally addressed by the algorithm, which treats movement decisions as discrete steps unrelated to the continuous nature of the robot's movement capabilities.

2. **Inability to leverage robot dynamics:** The A* algorithm here conducts a geometric, discrete search across the grid representation of the environment. It identifies waypoints that the robot must reach to progress towards its goal. However, this method does not incorporate the kinematic or dynamic aspects of the robot's motion—specifically, it does not consider velocities (first-order dynamics) or accelerations (second-order dynamics). By neglecting these factors, the paths generated by the algorithm are not only less efficient but also fail to take full advantage of the robot's capabilities to execute smooth and natural movements[34]. Consequently, while the path found might be technically feasible, it is not kinematically or dynamically optimal, nor is it smooth, which is the general expectation for social robots.

Kinodynamic Rapidly exploring Random Trees (Kinodynamic-RRT): The field of Sampling-based Motion Planning has come up with several robust alternatives for navigating complex environments, particularly for systems like holonomic robots. We implemented the Kinodynamic-RRT [35] algorithm, which is an approach that reasons about the system using kinematic and dynamic constraints. It is particularly effective in high-dimensional spaces or scenarios where obstacles are densely and unpredictably distributed.

Pseudo-code of the algorithm is presented in Algorithm 1. It works by incrementally building a tree that branches out from the starting node towards the goal. Points are randomly sampled in the environment, and the closest node in the tree is chosen. It then randomly samples the robot controls and propagates these controls. If the new position can be reached without colliding with obstacles, then the new position is added as a new node in the tree. The algorithm is also biased to explore towards the goal by choosing the goal position as the random position with some probability. A tree explored by the algorithm for the environment is illustrated in Figure 6.

Unlike the discrete path planning of A*, Kinodynamic-RRT considers the continuous nature of robot movement, effectively utilizing the robot's full range of motion capabilities, such as fluidly adjusting direction and optimizing travel paths in real-time. By dynamically adapting to the environment, the algorithm can often find paths more quickly and flexibly than grid-based methods.

Limitations of Kinodynamic-RRT:

1. **Continuous Movement and Stationary Strategy:** This approach continuously seeks new nodes to expand the tree, so it does not consider staying stationary. This continuous movement can be a drawback in environments with moving agents, as the algorithm's predisposition to always move can lead to it not finding viable paths in dynamic scenarios. This limitation can be addressed by incorporating a strategy where the robot temporarily remains stationary, allowing other moving agents to adjust their positions before resuming the search for a viable path.

2. **Randomness in Path Generation:** Due to the randomness in its path generation process, the paths

Algorithm 1 Kinodynamic-RRT

Initialize tree T with initial position x_{init}
while not reached goal **do**
  $x_{\text{rand}} \leftarrow \textsc{RandomState}()$
  With probability ϵ: $x_{\text{rand}} \leftarrow x_{\text{goal}}$
  $x_{\text{nearest}} \leftarrow \textsc{Nearest}(T, x_{\text{rand}})$
  $u_{\text{rand}} \leftarrow \textsc{RandomControls}()$
  $x_{\text{new}} \leftarrow \textsc{Steer}(x_{\text{nearest}}, u_{\text{rand}})$
  if $\textsc{NoCollision}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $T.\textsc{AddVertex}(x_{\text{new}})$
    $T.\textsc{AddEdge}(x_{\text{nearest}}, x_{\text{new}})$
  end if
  if $\textsc{ReachedGoal}(x_{\text{new}}, x_{\text{goal}})$ **then**
    return T
  end if
end while
return failure

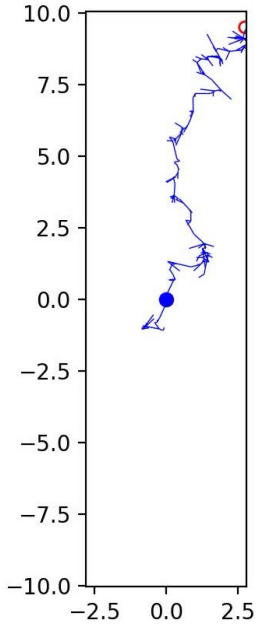


Figure 6: Tree explored by Kinodynamic-RRT. The blue dot represents the start position, and the red circle represents the goal region.

produced by the algorithm can sometimes be erratic and jagged, lacking the smoothness desired for efficient navigation. This randomness can lead to suboptimal paths that are longer or more convoluted than necessary. This limitation has been mitigated by more advanced versions of the algorithm, such as RRT* [36], which includes a path optimization process. However, these enhancements have not been investigated in the context of this work.

Reinforcement Learning (RL): In our simulation environment (described in Section 4.2.3), we experimented with reinforcement learning for robot navigation using the Soft Actor-Critic algorithm [37] combined with Hindsight Experience Replay [38]. This method was trained using sparse rewards, assigning a -1 penalty for every step the robot took without reaching the goal and a zero reward upon reaching the goal. Episodes were terminated either when the robot successfully reached its destination or collided with an obstacle, providing direct feedback on navigation efficacy and collision avoidance.

Limitations of RL:

1. **Limited Exploration Due to Collisions:** In scenarios where a robot must navigate without colliding with humans, penalties from such collisions lead to terminated training episodes. This severely restricts the algorithm's ability to explore the environment comprehensively, inhibiting its learning and adaptation capabilities. The reduced exploration prevents the algorithm from experiencing a wider array of states and actions necessary for developing effective long-horizon navigation strategies in dynamic settings.
2. **Inefficiency with Long-Horizon Problems:** Pure RL approaches often face challenges in long-horizon problems due to their reliance on sparse rewards and delayed feedback. This setup makes it difficult for the RL algorithm to link specific actions with their long-term outcomes, leading to inefficient learning

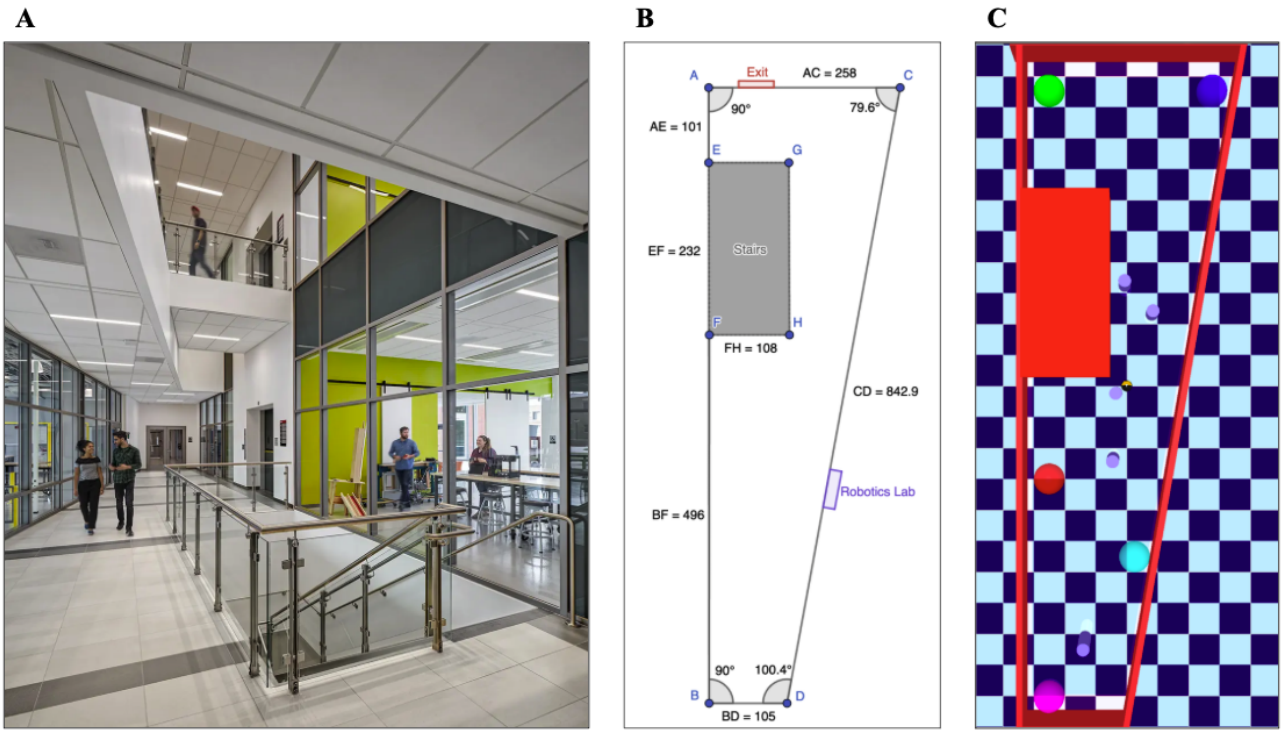


Figure 7: Simulating a real-world environment. (A) An academic building corridor was chosen as a test environment. (B) This environment's basic dimensions were measured to recreate in the MuJoCo simulation. (C) The simulation included not only corresponding dimensions and physical obstacles but also moving human obstacles in purple and a model of the robot in black and yellow.

paths and extended training periods.

4.2.3 Simulation

Simulation Software: Simulations play a crucial role in the development of robotics, particularly in the field of robot motion planning. They provide a controlled and repeatable environment where algorithms can be comprehensively tested and refined without the risks associated with real-world testing. This is especially valuable in developing learning-based methods, which often require extensive datasets to train effectively. Simulations offer a cost-effective solution by enabling the generation of large volumes of training data under varied conditions, thereby accelerating the development and optimization of navigation strategies tailored for social interaction in complex environments.

In the development of our simulation environment, we evaluated several physics simulation engines to ensure robust and reliable performance across various platforms. After this evaluation, we chose the MuJoCo physics engine due to its cross-platform compatibility. Unlike other engines we considered, such as Gazebo and NVIDIA IsaacGym, which heavily relied on GPU-intensive computations, MuJoCo demonstrated an advantage by maintaining high performance without an exclusive dependence on GPU resources.

Simulating the environment: A corridor in an academic building was chosen to simulate the real-world environment. We started by measuring the simulation space and building an environment resembling the

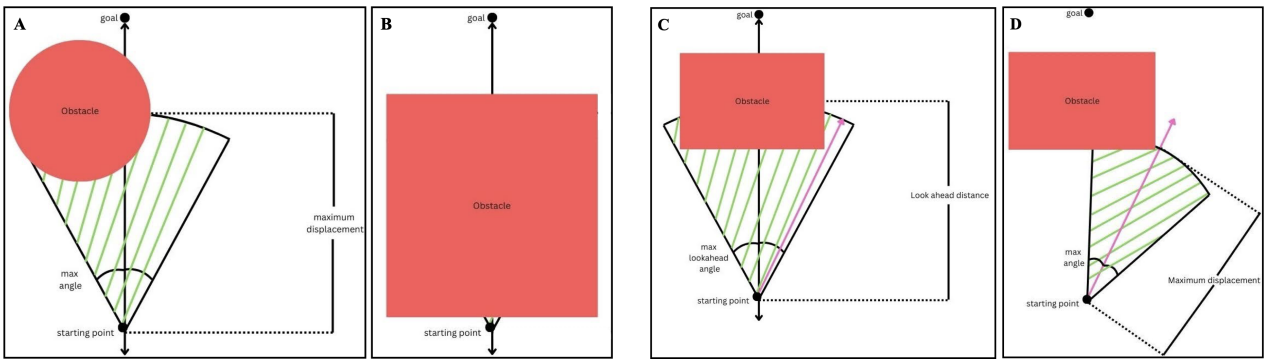
corridor in MuJoCo, with the dimensions and the angles aligning with the real-world measurements. Our next steps were importing a model of the robot in the MuJoCo environment and simulating human interactions. We built upon an open-source robot simulation model [39] to simulate the robot, its controls, and its movement. We also identified certain positions or sites (green = Werblin, blue = CoRE, red = BSC, pink = EE, teal = ARC) in the simulation environment to replicate Rutgers' Busch campus. These sites acted as starting or destination positions for the robot and the simulated humans.

Human Simulation: Accurately simulating humans is crucial for replicating real-world scenarios a robot may encounter despite the challenge of human unpredictability. We model humans as cylinders capable of omnidirectional movement in the simulation, as shown in Figure 7C. This movement characteristic adds flexibility in modeling their navigation behaviors and interactions within the environment. We simulate scenarios with up to 5 humans, where the simulation must effectively handle human navigation from a start point to a goal while avoiding collisions with the environment and other agents. This requires an efficient algorithm to support the simulation of multiple humans simultaneously.

Search-based cost-map Planner - A*: We implemented the optimized algorithm described in Section 4.2.2, which utilizes pre-computed cost-maps updated dynamically with the latest positions and velocities of the agents. While this approach is time-efficient, it demands substantial memory since it requires storing each agent's individual cost-map. This significant memory usage limits the method's viability to simulations involving no more than three humans. Although adjusting to a coarser grid granularity for the cost-maps could reduce memory requirements, it compromises the smoothness of the agents' movements, leading to unrealistic and jagged navigation patterns.

Optimal Reciprocal Collision Avoidance (ORCA): The ORCA algorithm [40] is utilized in robotics and simulations to orchestrate the movement of multiple agents within a shared space effectively. Initially, ORCA computes a goal vector indicating the preferred direction of movement. It then identifies "velocity obstacles" based on the positions and velocities of nearby agents, constructing a movement cone aligned with the goal vector that defines the maximum allowable displacement and deviation angle. The algorithm searches this cone for a feasible position that avoids collisions with the identified obstacles. This is illustrated in Figure 8(A). If no such position exists, the agent remains stationary, as it is deemed a safe position. This proactive strategy ensures collision-free movement for a predetermined future period, offering real-time responsiveness and scalability, which are crucial for managing dynamic environments.

However, ORCA is not without its drawbacks, particularly in its susceptibility to creating deadlocks. In scenarios where space is densely populated or dynamically complex, the algorithm might find a collision-free velocity that avoids immediate obstacles but subsequently leads to a situation where no agent can make an optimal move without risking a collision, as shown in Figure 8(B). This results in a deadlock, where the agents are stuck in sub-optimal positions with no viable path forward.



Optimal Reciprocal Collision Avoidance | Layered Optimal Reciprocal Collision Avoidance

Figure 8: Algorithms to simulate movement of humans. (A) The new position for a simulated human to move to is searched in a cone towards the goal. (B) In such a situation, the algorithm chooses to stay in position, leading to a local minimum. (C) To mitigate the local minima issue, a movement vector (in pink) is first searched in a large cone towards the goal vector. (D) Then, a smaller cone around the movement vector is used to search for the new position.

Layered Optimal Reciprocal Collision Avoidance: To address the deadlock issue, we developed the Layered ORCA algorithm, which enhances the basic ORCA by introducing a layered approach to obstacle avoidance. Initially, it selects a movement vector from the first level of ORCA with a larger cone, focusing solely on environmental obstacles without considering human-related obstacles. Once this movement vector is determined, a second ORCA calculation is performed, this time taking into account all types of obstacles, including both environmental and human. This method generally achieves efficient human navigation within our current simulation environment when the parameters are finely tuned. This multi-level strategy allows for more nuanced and responsive navigation tactics, better accommodating complex dynamic interactions within the simulated space.

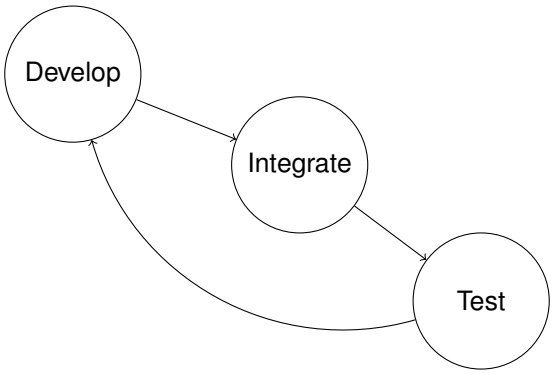
Even with the Layered ORCA approach, deadlocks can occur, although they are less frequent. To resolve these, we devised a strategy that involves having the deadlocked agents remain stationary for a pre-defined period. If the deadlock is resolved with the movement of other agents, the deadlocked agent can proceed toward their goal. If the deadlock is not resolved during this period, the goals of these agents are randomly reset. Subsequently, movement paths are changed, effectively resolving the deadlock and allowing navigation to proceed smoothly. This strategy effectively resolves deadlocks and can simulate scenarios where it appears that humans are meeting during the deadlock and dispersing, adding a layer of realistic social interaction to the simulation. Due to the Layered ORCA algorithm's computational efficiency and ability to effectively simulate human behavior, this approach was chosen to simulate humans.

The ORCA and Layered-ORCA algorithms, while effective for managing the movement of the simulated humans which are capable of instantaneous movement in any direction, pose challenges when applied to the LocoBot. The LocoBot requires an adjustment strategy to rotate in-place that accounts for its unique mobility capabilities. Effective implementation of ORCA requires the agent to move at any angle within the cone instantaneously. Therefore, while ORCA is apt for simulating human movements in this environment, it is less suited for directing the LocoBot. Therefore, in this work, ORCA was only utilized to simulate human behavior and not for robot navigation.

4.2.4 ROS Integration

Translating abstract implementation to real-world systems requires a careful understanding of the underlying mechanisms of the robot and what changes need to be made. Much of our approach focused on treating the individual components we developed as black boxes and then fitting these black boxes into the ROS ecosystem as a whole. Typically, this is done by creating a new node in ROS and then having a topic to publish or recieve data from. Therefore, the publication (pub) and subscription (sub) model is central to the integration problem as a whole. For example, one of the most critical nodes is the camera node, which publishes RGB data to a given topic. Other nodes will subscribe to this topic and use the RGB data as needed.

Overall, the ROS integration process is as follows:



Our methodology is based on the software development life cycle, which has similar steps, such as testing, development, and integration phases. Per the figure, our simplified setup has three steps: development, integration, and testing. Since we may need to redevelop our modules, this process is cyclical in nature. Following these steps, we were able to fix some of the bugs impacting the LocoBot, as well as integrate the new modules into the LocoBot.

5 Evaluations

Metric	**Mean ± Standard Error**
Success Rate of Accurate Transcription	0.4 ± 0.163
Success Rate of Accurate Answering	0.3 ± 0.1528
Latency (in seconds)	23.711 ± 1.199

Table 1: Evaluation of the Digital Interface with 10 trials.

The evaluations of the digital interface for(SPI) presented in Table 1 reveal areas for improvement. These metrics suggest moderate reliability in recognizing and responding to user input accurately. However, the biggest issue is parsing the proper location names around the campus. While Whisper seems to handle the bulk of the transcription without issues, names and nicknames like “Busch”, “Livingston,” or “College Ave” seem to cause problems. Additionally, the latency metric indicates a considerable delay in processing inputs. These results

underscore the need to refine the speech recognition and response mechanisms, particularly enhancing the fuzzy matching criteria and reducing response times to improve user experience and interface efficiency in navigating to desired locations. Whisper is a robust model with fine-tuning capabilities. So, if this project was scaled up to production status, a significant effort should be exerted to collect comprehensive data from the Rutgers population. Audio files with students uttering the names of locations with different accents should significantly improve the success rate. Additionally, the GPT module could also be fine-tuned to recover from transcription errors. Even in the current limited version, thanks to our system prompt, the GPT node was able to understand that transcription "living sun" meant "Livingston."

Scenario	**Metric**	**A***	**Kinodynamic-RRT**	**RL**
No Humans	Success Rate	**1 ± 0**	**1 ± 0**	0.04 ± 0.004
	Path Execution Time	100.48 ± 30.11	95.2 ± 37.8	**89.3 ± 10.28**
1 Human	Success Rate	**0.91 ± 0.06**	0.86 ± 0.09	0.021 ± 0.008
	Path Execution Time	109.48 ± 26.81	92.30 ± 37.6	**90.8 ± 12.84**
2 Humans	Success Rate	**0.87 ± 0.1**	0.7 ± 0.19	0
	Path Execution Time	114.69 ± 14.81	**95.342 ± 35.04**	-
3 Humans	Success Rate	**0.73 ± 0.15**	0.59 ± 0.11	0
	Path Execution Time	119 ± 20.94	**95.19 ± 38.59**	-
4 Humans	Success Rate	**0.64 ± 0.23**	0.58 ± 0.11	0
	Path Execution Time	121.46 ± 10.96	**96.238 ± 38.98**	-
5 Humans	Success Rate	**0.49 ± 0.16**	0.34 ± 0.21	0
	Path Execution Time	131 ± 7.62	**93.96 ± 36.90**	-

Table 2: Evaluation of the Robot Navigation Algorithms in the simulation environment. The results presented are means and standard errors collected from 10 runs per site, with the robot's goal set to one of five designated sites. The Path Execution Times are presented in seconds.

The Robot Navigation Algorithms presented in Section 4.2.2 were evaluated, and their results have been presented in Table 2. These algorithms were evaluated in the simulation environment where human agents were modeled using the Layered ORCA algorithm. To assess the performance of each navigation strategy, the simulation environment was populated with different numbers of humans, ranging from zero to five.

The formulation of the A* and RRT algorithms does not allow for collisions with humans. In the scenario where a collision is unavoidable, the A* algorithm chooses to stay in place, and the RRT algorithm cannot come up with a viable solution and, therefore, also stays in place. With the RL algorithm, however, collisions with humans are possible and such episodes are considered to be failures. For the purpose of these tests, any simulation episode that exceeded 140 seconds in duration was also classified as a failure based on the criterion that the robot did not reach its intended goal within the allotted time. This time constraint was applied to ensure that the navigation solutions not only avoided collisions but also achieved goals efficiently, reflecting

real-world conditions where timely navigation is often crucial.

For the A* algorithm, we used the optimized algorithm, which uses the pre-computed cost-maps. The A* algorithm maintains relatively high success rates across scenarios, decreasing as the number of humans increases, indicative of its robust path-finding capabilities even in more crowded environments. However, it generally exhibits longer path execution times than Kinodynamic-RRT, suggesting A* prioritizes safety over speed.

Kinodynamic-RRT, on the other hand, demonstrates a quicker path execution time in successful runs but suffers from a lower success rate, especially as the number of humans increases. This could be attributed to the algorithm's inability to halt, which is critical in dynamic environments to avoid collisions. A modified version of Kinodynamic-RRT that includes stopping capabilities could potentially offer a better balance between speed and reliability, addressing the current limitations in these scenarios.

RL shows notably poor performance, struggling to achieve satisfactory learning outcomes even in scenarios without humans. This indicates a significant challenge in using RL for this type of navigation task. This is possibly due to the sparsity of rewards and the complexity of the environment impacting the algorithm's ability to effectively explore the environment, which is crucial for RL algorithms in general. RL's inability to navigate effectively across all tested scenarios suggests that substantial modifications or alternative learning strategies may be necessary to enhance its applicability and success in similar environments.

6 Limitations

The current iteration of the robot is significantly scaled down relative to the planned deployment version, and any testing is conducted in controlled scenarios carried out primarily in simulation or a small area adjacent to the Robotics lab with the use of teleoperation. This poses a substantial hindrance since there is no way to truly gauge how our project will scale up to the ubiquitous complex indoor and outdoor environments on a college campus. The fragmented nature of the development process entails an extended integration of the pieces to work well together, avoiding conflicts between all the required packages and dependencies. The integration period led to the uncovering of unexpected conflicts, which required additional code rewrites and the development of brand-new solutions. The group's lack of bona fide roboticists made working with the fragile ROS structure challenging and susceptible to errors and missteps. Debugging and reverse engineering of the existing code to enhance our understanding accounted for at least as much time as was spent on developing and integrating the new code, making the process far from efficient.

The testing process was also limited in scope, as evaluations on metrics like path efficiency and energy usage on the real robot were not performed due to time and resource constraints. The lack of a fully functional navigational module forced us to rely exclusively on GPS coordinates for the Map Element and the GUI. While initially, we imagined the Map Element to be more of a hybrid between the OSM and the Robot's internal map; we had to shelve this idea until the navigational module worked adequately. However, the interactions between GPS and ROS coordinates have already been implemented, so once the prerequisites are completed, this part should be more of a fine-tuning rather than a full-blown development process.

7 Future Work

The limited time and resources heavily constrained the scope of integration and evaluations we could carry out. Considering the extensive range of new modules created for this iteration of the robot, it could be helpful to reimagine the flow of the whole system and optimize the already existing backend to accommodate the new elements. Expanding the evaluation framework to include energy efficiency, user satisfaction, and long-term reliability could provide deeper insights into the robot's performance. Additional testing under varying environmental conditions, such as weather changes and crowded settings, would help prepare the robot for practical deployment.

Another area for future work will be to evaluate alternatives for the campus guide robot's embodiment. In its desired state, the LocoBot should be able to perform various tasks, including seamless physical navigation. However, the robot's current design limits its physical and computational abilities. Future work should evaluate designs capable of handling all terrain on the Rutgers campus. This would mean moving away from the LocoBot and maybe looking into a more "rover" or car-like design. Additionally, the improved design should strive to balance being welcoming to a new and confused student and avoiding the uncanny valley effect. Notably, the new platform with increased computing power should allow us to make better use of machine learning tools and improve the quality of the experience that the robot can offer.

Improving the performance of the RL algorithm could be another key focus for future work, especially given its current limitations in handling complex navigation tasks. Implementing a dense reward system might offer the algorithm more frequent and informative feedback, potentially enhancing its ability to learn effective strategies more quickly. Additionally, using RL for short-horizon tasks specifically tailored to avoid humans while integrating a separate global planner for long-distance path-finding could optimize the strengths of RL in dynamic interactions and leverage more deterministic methods for overall route optimization. This hybrid approach could effectively combine the adaptability of RL in immediate, unpredictable scenarios with the reliability of traditional path-finding techniques for global route planning. Another critical area for future work involves implementing the real-time user tracking pipeline. Although designed and integrated conceptually, this functionality was not deployed onto the existing LoCoBot infrastructure due to time constraints. Closing the loop in this pipeline would allow the robot to ensure users are following along during navigation, enhancing its ability to mimic human guidance and improving overall engagement.

A final area for future work is integrating more human-in-the-loop procedures into the robot's training process. An obvious area for a human to be inserted is within the reinforcement learning-related aspects of the LocoBot. A human agent could examine the trajectories that the robot computed and correct them via a feedback mechanism. This human feedback could be worked into the reward functions for the training process. One could also consider using the end user's facial expressions and have the robot learn to use that data to improve the quality of its actions. This implicit feedback is useful as the robot can directly gauge its performance by estimating the human's mood without direct intervention. These two options are not the only way to incorporate human feedback; there are likely other valid ways to use this information. However, it is essential to remember that such solutions entail serious privacy concerns, so a rigid ethical framework should

also be developed in addition to the software.

Module	Metric	Setting
Digital Interface	Success Rate of Accurate information displayed	Only Interacting Person
Human Tracking	Percentage Accuracy of Inferred Positions using LiDAR	Single Obstacle-Human
	Percent Accuracy of Inferred Direction	Single Obstacle-Human
Real-Robot Navigation	Path Execution Time	0 to 5 Obstacle-Humans
	Success Rate of Reaching Destination	0 to 5 Obstacle-Humans
	Success Rate of Robot Stopping	Three people in view including the person interacting

Table 3: Suggested future evaluations for modules on the robot.

Table 3 presents suggested evaluations to advance the capabilities of different robot modules further. These assessments range from analyzing the digital interface's effectiveness to testing the precision of human tracking and the efficiency of real-robot navigation in environments with varying levels of complexity. Implementing these evaluations will provide valuable insights into the robot's performance and areas for improvement, ensuring more reliable and functional interactions in real-world applications.

8 Conclusion

The socially cognizant campus guide robot demonstrates meaningful progress toward creating a robotic system capable of navigating complex campus environments in a socially aware and accessible manner. By integrating technologies such as YOLOv8 for object detection, OC-SORT for human tracking, Whisper for speech recognition, and multimodal user interfaces, the robot can interact with users better and adapt to their needs. Several robot navigation algorithms have been developed and implemented, each with its own advantages and drawbacks, to address the specific needs of dynamic and crowded campus environments. Additionally, the development done in the simulation environments has increased the scope of building upon this system to test different navigation strategies in controlled, cost-effective ways.

Despite these advancements, the project has limitations, including restricted real-world testing and challenges in integrating various components into a cohesive system. Current evaluations highlight areas for improvement, such as reducing latency in the digital interface and increasing the reliability of speech recognition and navigation performance. Future work must address these challenges while expanding testing to diverse and more extensive environments.

Overall, this project has set up an infrastructure for improving upon the socially aware campus guide robot while emphasizing inclusivity, user engagement, and safe navigation.

References

[1] X.-T. Truong and T. D. Ngo, "Toward socially aware robot navigation in dynamic and crowded environments: A proactive social motion model," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 4, pp. 1743–1760, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8011466.

[2] A. of Robots, "Kar-go: The autonomous delivery vehicle for residential areas," 2019. [Online]. Available: https://www.academyofrobotics.co.uk/discover-kar-go.html.

[3] S. Technologies, "Starship technologies and ground delivery drones," 2016. [Online]. Available: https://www.nanalyze.com/2016/06/starship-ground-delivery-drones/.

[4] R. Murai, T. Sakai, H. Kawano, *et al.*, "A novel visible light communication system for enhanced control of autonomous delivery robots in a hospital," in *2012 IEEE/SICE International Symposium on System Integration (SII)*, IEEE, 2012, pp. 510–516. [Online]. Available: https://ieeexplore.ieee.org/document/6427311.

[5] A. Vega-Magro, R. Gondkar, L. J. Manso, and P. Núñez, "Towards efficient human-robot cooperation for socially-aware robot navigation in human-populated environments: The snape framework," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 3169–3174. [Online]. Available: https://ieeexplore.ieee.org/document/9561448.

[6] K. J. Dana, C. Andrews, K. Bekris, *et al.*, "Socially cognizant robotics for a technology enhanced society," *arXiv preprint arXiv:2310.18303*, 2023. [Online]. Available: https://ar5iv.labs.arxiv.org/html/2310.18303.

[7] A. Schneider, A. Robinson, C. Grimm, and N. T. Fitter, "How do starship robots affect everyday campus life? an exploratory posting board analysis and interview-based study," in *2024 33rd IEEE International Conference on Robot and Human Interactive Communication (ROMAN)*, IEEE, 2024, pp. 528–534. [Online]. Available: https://www.semanticscholar.org/paper/How-Do-Starship-Robots-Affect-Everyday-Campus-Life-Schneider-Robinson/560e6f1cf2b4ab0d349a049

[8] C. Spars, "Kiwi bot: An introduction to the robot-angel of food delivery," *UWIRE Text*, pp. 1–1, 2018. [Online]. Available: https://go.gale.com/ps/i.do?p=AONE&sw=w&issn=&v=2.1&it=r&id=GALE%7CA528499445&sid=googleScholar&linkaccess=abs&userGroupName=anon%7Ee9190957&aty=open-web-entry.

[9] S. Cai, A. Ram, Z. Gou, *et al.*, "Navigating real-world challenges: A quadruped robot guiding system for visually impaired people in diverse environments," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–18. [Online]. Available: https://dl.acm.org/doi/10.1145/3613904.3642227.

[10] F. Bu, A. W. Bremers, M. Colley, and W. Ju, "Field notes on deploying research robots in public spaces," in *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–6. [Online]. Available: https://arxiv.org/html/2404.18375v1.

[11] Y. Gao and C.-M. Huang, "Evaluation of socially-aware robot navigation," *Frontiers in Robotics and AI*, vol. 8, Article 721317, 2022. [Online]. Available: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2021.721317/full.

[12] C. Graham and N. Roll, "Evaluating OpenAI's Whisper ASR: Performance analysis across diverse accents and speaker traits," *JASA Express Letters*, vol. 4, no. 2, p. 025 206, Feb. 2024, ISSN: 2691-1191. DOI: 10.1121/10.0024876. eprint: https://pubs.aip.org/asa/jel/article-pdf/doi/10.1121/10.0024876/19692982/025206_1_10.0024876.pdf. [Online]. Available: https://doi.org/10.1121/10.0024876.

[13] H. Yadav and S. Sitaram, "A survey of multilingual models for automatic speech recognition," *arXiv preprint arXiv:2202.12576*, 2022. [Online]. Available: https://arxiv.org/abs/2202.12576?utm_source=chatgpt.com.

[14] T. B. Brown, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017. [Online]. Available: https://arxiv.org/abs/1707.06347.

[16] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, PMLR, 2018, pp. 1861–1870. [Online]. Available: https://proceedings.mlr.press/v80/haarnoja18b/haarnoja18b.pdf.

[17] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:14744621.

[18] H. Jiang, H. Wang, W.-Y. Yau, and K.-W. Wan, "A brief survey: Deep reinforcement learning in mobile robot navigation," in *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2020, pp. 592–597. DOI: 10.1109/ICIEA48937.2020.9248288.

[19] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[20] Z. Xu, Y. Li, X. Yang, Z. Zhao, L. Zhuang, and J. Zhao, "Open-source reinforcement learning environments implemented in mujoco with franka manipulator," in *2024 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, IEEE, 2024, pp. 709–714. [Online]. Available: https://arxiv.org/abs/2312.13788?utm_source=chatgpt.com.

[21] M. El-Hariry, A. Richard, and M. Olivares-Mendez, "Rans: Highly-parallelised simulator for reinforcement learning based autonomous navigating spacecrafts," *arXiv preprint arXiv:2310.07393*, 2023. [Online]. Available: https://arxiv.org/abs/2310.07393?utm_source=chatgpt.com.

[22] S. Wang and H. I. Christensen, "Tritonbot: First lessons learned from deployment of a long-term autonomy tour guide robot," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, IEEE, 2018, pp. 158–165. [Online]. Available: https://cseweb.ucsd.edu/~shengye/resources/wang2018tritonbot.pdf.

[23] B. Greenberg, D. Nakhimovich, R. Magnotti, *et al.*, "Development of a socially cognizant robotic campus guide," in *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, 2024, pp. 1229–1232.

[24] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, *Robust speech recognition via large-scale weak supervision*, 2022. arXiv: 2212.04356 [eess.AS]. [Online]. Available: https://arxiv.org/abs/2212.04356.

[25] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding with unsupervised learning," 2018. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[26] OpenStreetMap contributors, *Planet dump retrieved from https://planet.osm.org*, https://www.openstreetmap.org, 2017.

[27] D. Luxen and C. Vetter, "Real-time routing with openstreetmap data," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '11, Chicago, Illinois: Association for Computing Machinery, 2011, pp. 513–516, ISBN:

9781450310314. DOI: 10.1145/2093973.2094062. [Online]. Available: https://doi.org/10.1145/2093973.2094062.

[28] J. Cao, J. Pang, X. Weng, R. Khirodkar, and K. Kitani, "Observation-centric sort: Rethinking sort for robust multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9686–9696. [Online]. Available: https://arxiv.org/abs/2203.14360.

[29] Ultralytics, *Ultralytics yolov8 docs*. [Online]. Available: https://docs.ultralytics.com/.

[30] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960. [Online]. Available: https://asmedigitalcollection.asme.org/fluidsengineering/article-abstract/82/1/35/397706/A-New-Approach-to-Linear-Filtering-and-Prediction?redirectedFrom=fulltext.

[31] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection (2015)," *arXiv preprint arXiv:1506.02640*, vol. 825, 2015. [Online]. Available: https://arxiv.org/abs/1506.02640.

[32] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. DOI: 10.1109/tssc.1968.300136. [Online]. Available: https://doi.org/10.1109/tssc.1968.300136.

[33] T. Frankel, "Holonomic and nonholonomic constraintsholonomic and nonholonomic constraints," in *The Geometry of Physics: An Introduction*. Cambridge University Press, 2011, pp. 165–188.

[34] E. Schmerling and M. Pavone, "Kinodynamic planning," *Encyclopedia of Robotics, Springer*, 2019.

[35] D. J. Webb and J. van den Berg, *Kinodynamic RRT*: Optimal Motion Planning for Systems with Linear Differential Constraints*, 2012. arXiv: 1205.5088 [cs.RO]. [Online]. Available: https://arxiv.org/abs/1205.5088.

[36] K. Naderi, J. Rajamäki, and P. Hämäläinen, "RT-RRT*: A real-time path planning algorithm based on rrt," in *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, 2015, pp. 113–118.

[37] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement," in *Proceedings of the 35th International Conference on Machine Learning. July 10th-15th, Stockholm, Sweden*, vol. 1870, 2018.

[38] M. Andrychowicz, F. Wolski, A. Ray, *et al.*, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[39] *PyRobot-RL*, `https://github.com/Tiga002/pyrobot-rl`, 2020.

[40] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 3–19, ISBN: 978-3-642-19457-3.