UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Technology
Robotics and Computer Engineering

Paola Avalos Conchas

Socially Aware Planning for Indoor Navigation

Master's Thesis (30 ECTS)

Supervisor(s): Bin Zhang, Ph.D.
Kanagawa University
Karl Kruusamäe, Ph.D.
University of Tartu

Tartu 2025

Socially Aware Planning for Indoor Navigation

Abstract: As robots increasingly present in human-populated spaces, they must be able to navigate among humans safely and without disrupting. People try to preserve their own personal space when moving in real-world social spaces. However, the current navigation methods do not consider this aspect and treat humans as any other obstacle.This thesis proposes a method that considers personal space for humans as well as social navigation norms. For this purpose, the robot converts camera-based human detections to a costmap form and define the personal space as a Gaussian asymmetric function. The proposed solution is validated through real-world experiments, demonstrating that the robot can improve the quality of navigation. The proposed solution is available on GitHub as a costmap layer that can be easily integrated into existing frameworks.

Keywords:
Dynamic avoidance, robotics, ROS, autonomous navigation, mobile robot

CERCS: T125 Automation, robotics, control engineering.

Sotsiaalselt teadlik planeerimine siseruumides navigeerimiseks

Lühikokkuvõte: Kuna roboteid on üha enam inimestega asustatud ruumides, peavad nad suutma inimeste seas turvaliselt ja segamatult liikuda. Inimesed püüavad reaalsetes sotsiaalsetes ruumides liikudes säilitada oma isiklikku ruumi. Praegused navigatsioonimeetodid aga seda aspekti ei arvesta ja kohtlevad inimesi kui mingit muud takistust. Käesolev lõputöö pakub välja meetodi, mis arvestab nii inimeste isiklikku ruumi kui ka sotsiaalsete navigatsiooninormidega. Sel eesmärgil teisendab robot kaamerapõhised inimtuvastused kulukaardi vormiks ja defineerib isikliku ruumi Gaussi asümmeetrilise funktsioonina. Pakutud lahendust valideeritakse reaalsete katsete abil, mis näitavad, et robot suudab navigatsiooni kvaliteeti parandada. Pakutud lahendus on saadaval GitHubis kulukaardi kihina, mida saab hõlpsasti integreerida olemasolevatesse raamistikesse.

Võtmesõnad:
Dünaamiline vältimine, robootika, ROS, autonoomne navigatsioon, mobiilrobot

CERCS: T125 Automatiseerimine, robootika, juhtimistehnika

Abbreviations

- AI - Artificial Intelligence
- API - Application Programming Interface
- ASR - Automatic Speech Recognition
- CPU - Central Processing Unit
- DWA - Dynamic Window Approach
- GPU - Graphics Processing Unit
- LiDAR - LIght Detection And Ranging
- LLM - Large Language Model
- NLP - Natural Language Processing
- OS - Operating System
- RGB-D - Red Green Blue – Depth
- RL - Reinforcement Learning
- ROS - Robot Operating System
- YOLO - You Only Look Once (algorithm)

Contents

1 Introduction

1.1 Background

Mobile robots are increasingly operating in human-populated spaces, from warehouses to hospitals, reflecting the growing recognition of their potential to offer productivity gains and operational support [1]. However, deploying robots in shared spaces requires safety guarantees to ensure they do not endanger humans or disrupt workflows. The safe integration of mobile robots hinges on their ability to navigate between static infrastructure and unpredictable dynamic obstacles (e.g., humans, other robots) without causing disruptions or hazards. Current open-source navigation stacks in ROS provide basic obstacle avoidance capabilities, but often lack handling of social navigation norms and dynamic human movements.

1.2 Motivation

While ROS provides comprehensive tools for mobile robot navigation through its Navigation stack, there remains a significant gap in handling dynamic human environments effectively. Existing solutions typically treat humans as simple obstacles without considering social navigation norms or predictive behaviors. Moreover, the integration of real-time human detection data from traditional LiDAR-based systems is often suboptimal. With mobile robots becoming more prevalent in human spaces, there is a pressing need to enhance ROS Navigation's capabilities to include socially-aware motion planning and improved dynamic obstacle representation.

1.3 Objectives and Contributions

The objective of this thesis is to enhance dynamic obstacle avoidance for mobile robots in human environments by combining real-time sensor fusion and adaptive cost mapping. The contributions of this work are:

- **C1:** A custom costmap layer that integrates real-time human pose data, improving dynamic obstacle representation. This adaptation of human detection software works in the context of ROS and provides more accurate environment perception.
- **C2:** An adaptive trajectory planner that balances collision avoidance with social navigation norms. This includes modifications to DWA local planners to take social navigation into account and the development of pre-existing trajectory prediction implementations for use in planners.

- **C3:** Experimental validation in a real environment, demonstrating an improvement in dynamic obstacle avoidance compared to baseline ROS navigation stacks while maintaining navigation efficiency.

It is worth noting that while the developed solutions are implemented and tested on a specific robotic platform, the approaches are designed to be generalizable to various mobile robot configurations operating in human indoor environments.

1.4 Organization of Thesis

The thesis is organized as follows:

- **Introduction:** Introduction, background, motivation, and contributions of the thesis are discussed.
- **Background:** This chapter presents a literature review on autonomous navigation, the challenges of dynamic obstacle avoidance, and introduces detection methods and related works.
- **Methodology:** Detailed explanation of the proposed solutions, including the custom costmap layer, intent extraction from speech and adaptive trajectory planner.
- **Results:** Findings from experimental validation in real environments, with comparisons to baseline approaches.
- **Discussion:** Discussing the results and limitations of the thesis.
- **Conclusion:** Summary of outcomes and contributions of the work.

2 Background

This chapter establishes foundational concepts discussed in this thesis for robotic navigation in dynamic human environments. It analyzes technical challenges from real-time perception to path planning, while exploring solutions developed across sensing, planning, and prediction domains.

2.1 Autonomous Navigation

Autonomous navigation enables vehicles or robots to operate independently by combining environmental understanding, decision-making, and precise motion execution. It enables robots to move independently by integrating four core capabilities defined by Siegwart et al. [2]:

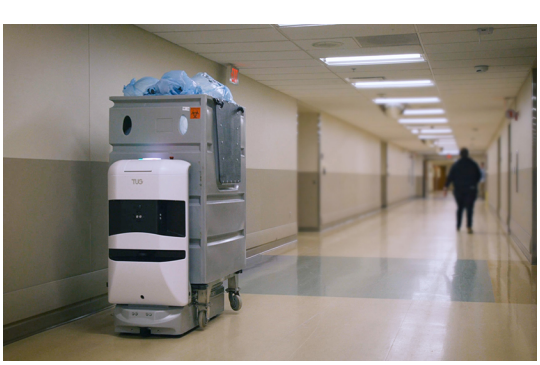
1. **Perception:** Interpreting sensor data (LiDAR, cameras, etc.) to construct real-time environmental representations.
2. **Localization:** Estimating the robot's position within these representations using techniques like SLAM.
3. **Cognition:** Generating actionable goals through dynamic path planning and obstacle avoidance.
4. **Motion Control:** Executing physical movements via controllers that translate plans into wheel or actuator commands.

These pillars operate together. Perception systems first build maps by fusing data from sensors, creating a foundation for localization algorithms to track the robot's position. Cognition then leverages these spatial models to plan paths that balance efficiency with safety, employing global planners for coarse route optimization and local planners for real-time obstacle avoidance. For instance, Starship Technologies' delivery robots precompute campus-wide routes but dynamically adjust paths to avoid pedestrians [3], demonstrating this layered planning hierarchy.

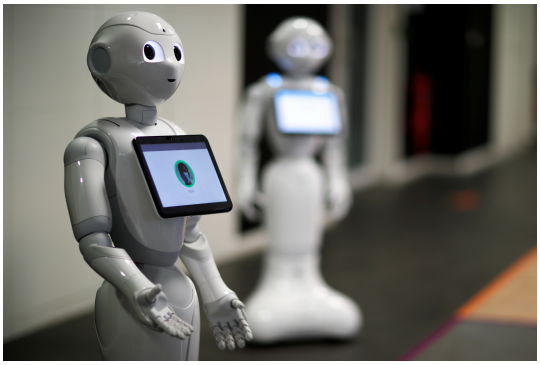
Motion control closes the loop by executing trajectories with precision. Controllers such as PID or model-predictive systems account for wheel slippage and terrain variations, ensuring adherence to planned paths. Simultaneously, reactive obstacle avoidance algorithms continuously monitor proximity sensors to maintain safe distances from both static infrastructure and dynamic agents like humans.

Real-world implementations highlight this integration. Amazon's Kiva robots navigate warehouses by dynamically updating routes around mobile workers and stationary shelves [4], while Aethon's TUG hospital robots adapt to crowded medical corridors using socially aware navigation protocols [5]. Retail robots like SoftBank's Pepper

further illustrate the role of cognition, combining customer interaction with adaptive pathfinding in unstructured environments [6]. Ultimately, autonomous navigation hinges on the seamless interaction of perception, localization, cognition, and control—each component iteratively refining actions to achieve safe, goal-oriented mobility. Figure 1 shows two representative platforms used in healthcare and retail environments.



(a) *Aethon TUG in hospital corridors*



(b) *SoftBank Pepper in retail environments*

Figure 1. *Examples of service robots: (a) Aethon TUG [7]; (b) SoftBank Pepper [8].*

2.2 Dynamic Avoidance

The field of robotic navigation has evolved significantly from simple static obstacle avoidance to the complex challenge of dynamic obstacle avoidance, which has become increasingly critical as robots move into human-populated environments [9] [10]. While static navigation deals with predictable, unchanging obstacles like walls and furniture, avoidance of objects in motion, "dynamic obstacles", is much more complex as it must account for people, vehicles, and other robots - each with their own unpredictable behaviors and trajectories [11]. This fundamental difference creates numerous technical challenges that span across perception, prediction and planning.

Mobile robots find it difficult to navigate autonomously in a dynamic environment since most traditional algorithms are only efficient in a static environment. In such an environment, a mobile robot usually works based on mapped information on the existing obstacles. So, it becomes a major problem for the mobile robot to autonomously navigate in a dynamic environment as it is not equipped to learn to plan its path to avoid dynamic obstacles continuously [12]. The two main challenges traditional methods face are the contradiction between the accuracy of grid-based map representation and its memory requirements and the fact that, in dynamic environments, these algorithms require intensive calculations for real-time re-planning of the navigation path making their reactivity is somewhat limited [13].

Modern perception systems must resolve the tension between latency and reactivity.

Boston Dynamics' *Stretch* robot exemplifies this balance, leveraging fused LiDAR and depth camera data to classify warehouse workers as dynamic obstacles with sub-second latency, such systems increasingly employ temporal differencing techniques to distinguish static and dynamic elements in point clouds, though occlusions and sensor noise remain persistent challenges [14]. Accurate prediction is equally critical, requiring robots to anticipate both movement patterns and human social behavior. At the planning level, traditional grid-based methods like A* become computationally prohibitive when frequent replanning is required. This has spurred adoption of reactive techniques such as DWA and Velocity Obstacles, which enable local adjustments without global recomputation. Amazon's *Proteus* warehouse robots apply this principle, combining coarse A* waypoints with real-time DWA corrections to navigate around forklifts while maintaining throughput targets [15].However, these methods still struggle with long-horizon reasoning, often failing to anticipate cascading interactions in dense scenarios.

2.3 Robot Operating System

Robot Operating System (ROS) is an open source, Linux-based framework to operate and program various kinds of robots. It leverages peer-to-peer communication, in which executable programs, called nodes, communicate with each other at runtime [16]. The nodes are connected to a ROS master, which is in charge of coordinating the communication among them and through which they can exchange information. The nodes communicate through publishing or subscribing to topics containing messages. Therefore, in case a node needs some data, it subscribes to the relevant topic and, likewise, if it generates some data, it publishes them in form of messages on a topic. ROS makes the robot system decoupled, so that different parts of robots can perform different functions without hindering the others, thus if one of the nodes crashes, it does not necessarily halt the system entirely. Another advantage of ROS is reusability of the code on various robots, as it is hardware agnostic.

2.3.1 ROS Navigation

ROS Navigation is a framework that exposes functionalities of trajectory planners through a ROS based interface, using a collection of packages that enable mobile robots to move in the environment avoiding obstacles encountered along the way from its current position to a goal position. It is designed to be as general-purpose as possible primarily meant for differential drive (two separately driven wheels placed on either side of the robot body) and holonomic wheeled robots (robots can move in any direction) [17].

The core of the Navigation framework is the move_base node seen in Fig. 2, which takes a goal pose as input and generates a trajectory for the robot to follow from its current position to the goal position. The move-base block links the global and local planners to adjust the behavior of the robot during path planning. The robot's position

is estimated using odometry, which uses motion sensors like encoders, and the data is published on the /odom topic.

The output of the path planning is given in the ROS topic /cmd_vel as a geometry_msgs/Twist message, which contains linear and angular velocity vectors. The robot base controller then interprets these vectors and converts them into the corresponding motor commands to follow the desired trajectory.

In order to receive goal requests, ROS Navigation uses ActionServer for communication with an ActionClient. In ROS, services follow a client-server model. A node that wants to utilize a specific service acts as a client and sends a request to the node that provides the service, which acts as the server. The server processes the request and sends back a response to the client. Move-base is an implementation of a SimpleActionServer, which has a single goal policy, subscribed to the topic move-base-simple/goal. A SimpleActionClient can send goal poses to the server, and move-base generates a trajectory making use of global and local planners.

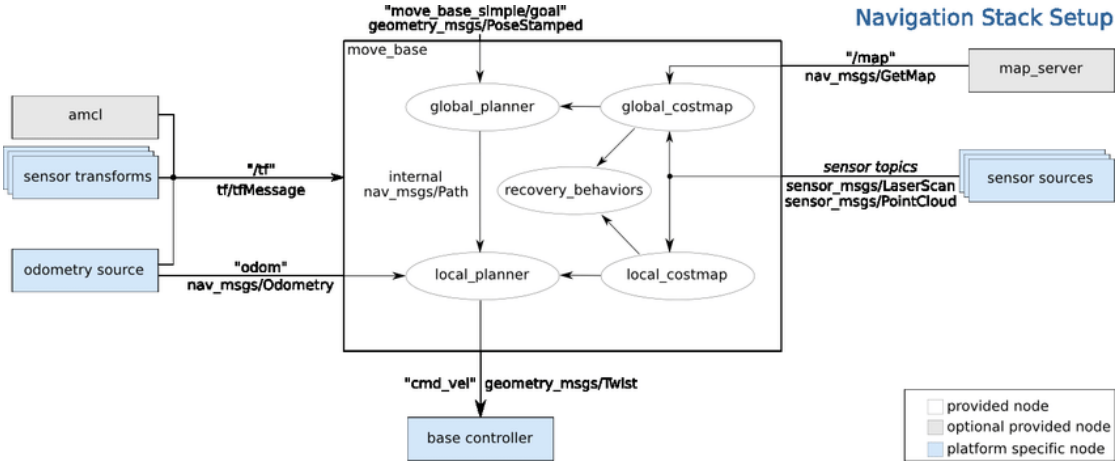


Figure 2. *Structure of ROS Navigation, which integrates localization, mapping, global and local planners.*

2.3.2 Costmap Representation in ROS

The fundamental task of autonomous real-world navigation is to reach a global goal (global navigation) while maneuvering in a dynamic environment where it is necessary to react to static and dynamic obstacles (local obstacle avoidance) [1]. Central to this process in the ROS Navigation framework is the costmap, a grid-based representation of the environment where each cell encodes a "cost" value reflecting traversal risk (e.g., obstacles, proximity penalties). Unlike traditional monolithic costmaps, where all the data are stored in a singular grid of values, in the costmap layers approach used in ROS, each layer tracks one type of obstacle or constraint, and then modifies a master costmap

that is used for the path planning. The base layers in ROS Navigation are essentially three: (i) static layer – stores the costs associated with the static map provided at launch time, (ii) obstacle layer – continuously marks and clears cells according to sensor data, and (iii) inflation layer – propagates cost values out from occupied cells that decrease with distance, in order to provide a safety margin for the robot navigation [18].

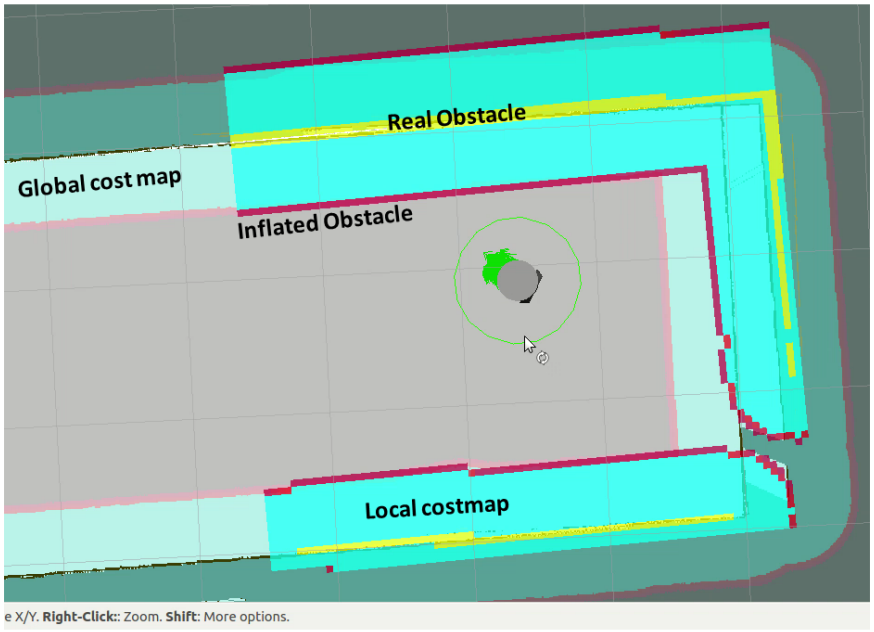


Figure 3. *Layered-costmap representation in ROS*

2.3.3 Global and Local Planners

ROS Navigation uses two-level planning - global and local - for calculating the series of velocity commands from the robot's current pose to its goal state [19]. After perceiving the environment using sensors, it is necessary to plan a feasible path in order to reach the desired target starting from the current position. There is a rich literature on path planning algorithms which can be categorized to traditional methods (A*, Dijkstra, etc.), AI-based methods (RL, genetic algorithm, etc.), and hybrid methods which combine traditional approaches with AI methods [20] [21].

In global navigation, traditional planners like A* or Dijkstra's algorithm compute a complete path using the static layer of the costmap. These algorithms prioritize low-cost cells to generate optimal routes while avoiding high-cost regions such as walls. For dynamic environments, the obstacle layer continuously modifies the costmap with

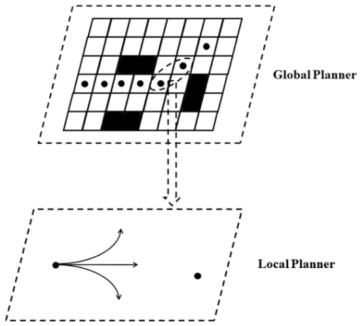


Figure 4. *Global and Local planner in ROS*

sensor data, enabling the local planner to adjust trajectories in real time [22]. Local planning leverages the full costmap (static + obstacle + inflation layers) to refine the global path. Reactive algorithms like the DWA or Vector Field Histogram evaluate the costmap's cell values to compute collision-free velocities that respect kinematic constraints. The inflation layer ensures smooth detours around obstacles by penalizing paths near high-cost regions, while sensor data updates the obstacle layer to reflect moving objects.

A global planner takes into account the base footprint, a projection of the robot's outline on the ground plane, and a global costmap to generate a collision-free trajectory consisting of series of wayposes to reach the goal state. However, for the sake of simplicity and efficiency, global planners consider the base footprint as a single point in the process of generating the global plan, if the base footprint is defined as a polygon geometry. Since mobile platforms are commonly navigated by velocity commands, local planners in ROS Navigation are in charge of translating the wayposes provided by the global planner into corresponding velocity commands to be sent to the robot [19]. Although global planners publish the whole trajectory to be followed by the robot, local planners, too, plan for some variable steps ahead at each instance of time and update the global trajectory. This way, local planners would be able to fix any errors due to slipping and driving precision while trying to follow the global planner's trajectory. Thus, this two-level planning increases the planning accuracy and helps avoiding any new appearing or moving obstacles. However, this also means the trajectory generated by local planners is not necessarily following the initial global plan [23].

2.4 Detection Methods

Accurate obstacle detection forms the foundation of safe autonomous navigation, requiring systems to interpret diverse sensor inputs under dynamic conditions. This section outlines key detection modalities and their integration:

- **LIDAR/Radar:** Provide reliable distance measurements and object detection across lighting conditions.
- **Vision-Based Systems:** Enable semantic understanding through cameras and computer vision.
- **Sensor Fusion:** Enhance robustness by combining complementary sensor data streams.

2.4.1 LiDAR

LiDAR (Light Detection and Ranging) has become a cornerstone of autonomous navigation due to its high-resolution 3D mapping capabilities. By emitting laser pulses and measuring their time-of-flight, LiDAR generates precise point clouds that capture environmental geometry with centimeter-level accuracy. Recent advancements in solid-state LiDAR have improved reliability while reducing cost and mechanical complexity, enabling widespread adoption in robotics and autonomous vehicles. For example, Waymo's autonomous fleet relies on multi-beam LiDAR arrays to detect and classify obstacles at ranges exceeding 200 meters, even in low-visibility conditions [24] .Similarly, research by Malavazi et al. [25] shows LiDAR-only system applications in crop maintenance,while Wang et al. [26] show a complete navigation system proposed for mobile ground vehicles in a park environment using a LiDAR as the only sensor.

2.4.2 Camera-Based Detection

Vision-based systems have gained prominence with the rise of deep learning, enabling robots to interpret complex scenes semantically. Tesla's "Vision-only" Autopilot system exemplifies this trend, leveraging a surround-camera array and neural networks to achieve real-time object detection, lane estimation, and path prediction without LiDAR . Their approach relies on massive datasets and iterative training to handle edge cases—such as occluded pedestrians or irregular road geometries—demonstrating the scalability of pure vision systems in structured environments [27].

Vision systems excel at identifying obstacles through semantic context and Depth-enabled cameras like the Intel RealSense further enhance spatial reasoning by converting 2D detections into 3D coordinates, stereo vision and structured light technologies enable precise localization—critical for collision avoidance in cluttered spaces. A representative

approach by Joul [28] combines RGB-D cameras with YOLO, a real-time object detection network. The pipeline processes RGB images with YOLO to generate 2D bounding boxes, then merges these detections with depth maps to estimate 3D human positions. This hybrid approach balances speed (30 FPS on mid-tier GPUs) and accuracy, enabling reliable human tracking in dynamic settings.

Modern systems increasingly leverage such vision-depth fusion, as cameras provide rich environmental context while depth sensors resolve scale ambiguity—critical for navigation tasks requiring spatial reasoning. Depth-enabled cameras like the Intel RealSense convert 2D detections into 3D coordinates through geometric deprojection. By combining pixel locations with depth data and camera intrinsics (focal lengths, optical centers), systems map image coordinates to real-world positions. This process allows robots to perceive obstacle geometry directly in their operational coordinate frame. Depth disparity from stereo vision enables precise spatial localization—a capability fundamental for collision prediction and avoidance.

2.4.3 Sensor Fusion

Effective navigation in dynamic environments requires perception systems that overcome the limitations of individual sensors through complementary fusion. Traditional 2D LiDAR provides reliable mapping of static structures but struggles with dynamic obstacle classification and suffers from background clutter interference at operational frame rates [29, 12]. Cameras offer superior object recognition through rich visual features, yet lack inherent depth perception.

Modern systems address these challenges through tightly-coupled sensor fusion architectures that combine the strengths of each modality. LiDAR delivers precise, long-range 3D measurements while cameras provide crucial semantic context, enabling systems to distinguish between movable objects and fixed infrastructure, as demonstrated by Pan et al. [30], by correlating geometric point clouds with visual features resolves ambiguities in crowded environments where single-sensor approaches fail. Wu et al. [31] further show how such fusion improves both obstacle detection reliability and tracking performance compared to standalone sensors.

The synergy between these sensors enables robust real-time navigation. LiDAR supplies high-frequency spatial updates while cameras validate obstacle identities, creating probabilistic representations that account for both immediate collisions and anticipated movements. This multi-modal approach proves particularly valuable for dynamic obstacle avoidance, where maintaining accurate tracking of moving objects is essential for safe path re-planning. The sparse but precise 3D data from LiDAR complements the dense but geometrically ambiguous camera data, yielding a perception system greater than the sum of its parts [30].

2.5 Related Works

Socially-aware navigation has become an increasingly active area of research in robotics, particularly in indoor environments where mobile robots share space with humans. Several approaches have focused on extending standard navigation to support human-aware path planning. A common technique involves augmenting the costmap with social constraints. For example, the *nav2_social_costmap_plugin*[32] introduces a ROS2-compatible plugin that inflates costs around humans using asymmetric Gaussian distributions based on the detected person's motion and velocity. Similarly, Barış [33] developed a social costmap layer for ROS1 using laser-based leg detection to enforce proxemic boundaries and evaluate navigation behavior via a Human Comfortable Safety Index.

Recent work by Pérez et al. [34] and Ribeiro & Moreno [35] extends the concept of social costmaps by incorporating human motion prediction and group-aware navigation. These methods use anticipatory models to adjust the robot's path dynamically, improving human comfort and reducing abrupt trajectory changes.

In terms of local planning, adaptive variants of the Dynamic Window Approach (DWA) have been proposed. Dobrevski and Škocaj[36] introduced an ADWA model that uses learned cost function weights to adapt robot behavior in cluttered environments. Ngo et al.[37] further optimized DWA to incorporate personal space constraints, making it more suitable for socially sensitive tasks.

Camera-based human detection pipelines have also become more prominent with the use of deep learning. Joul's ROS-compatible YOLO-based detector [28] provides real-time 3D bounding boxes that can be directly integrated into costmap layers. Similarly, Kang et al. [38] fuse RGB-D data to detect humans, considers different social types and defines appropriate personal spaces for each.

2.6 Intent Detection

Voice-based human-machine interface has become a prevalent feature for modern intelligent vehicles, especially in navigation and infotainment applications [39]. Automatic Speech Recognition (ASR) converts spoken audio streams to plain texts, but a follow-up Natural Language Processing (NLP) sub-system is needed to understand the contextual meaning from the text and act. For the specific human-robot navigation dialogue application, the two major tasks include (1) intent detection - decide whether a sentence is navigation-related, and (2) semantic parsing - retrieve important information (e.g., point-of-interest destinations) from the words [39]. This capability becomes particularly crucial in mobile robotics applications where voice serves as the primary interface for commanding autonomous navigation in complex environments.

Traditional approaches to navigation intent detection have employed various methodologies with differing levels of sophistication. Rule-based systems utilize predefined

grammars and pattern matching to identify command structures, such as parsing phrases following the template "[Action] to [Location]" (e.g., "Go to the kitchen"). While effective for constrained vocabularies, these systems demonstrate limited flexibility when confronted with the natural variation in human speech. Statistical methods, particularly those leveraging machine learning classifiers such as Support Vector Machines (SVMs) or neural networks, give improved robustness by learning intent patterns from labeled datasets of navigation commands. These models treat intent detection as a classification problem, differencing navigation commands from other speech acts while simultaneously identifying key command components through sequence labeling techniques.

The emergence of Large Language Models (LLMs) such as GPT-3.5 and GPT-4 has introduced transformative capabilities to navigation intent detection. These models excel at interpreting free-form navigation commands without relying on rigid templates, enabling comprehension of varied phrasings including indirect requests (e.g., "Could we move toward the exit?") or implied destinations (e.g., "I need to get my lunch" when near a refrigerator). LLMs can filter non-navigation speech, resolve referential ambiguities using contextual cues, and output structured navigation goals compatible with robotic systems.

Several studies have demonstrated the effectiveness of LLM-based approaches in robotic navigation scenarios. Huang et al. [40] implemented a GPT-3.5 powered intent detection system capable of interpreting navigation commands across multiple languages, significantly outperforming traditional classifiers in handling paraphrased commands. Similarly, Liao et al. [41] developed a multimodal intent detection framework combining GPT-4 with visual cues, enabling resolution of ambiguous references (e.g., "Go there") through gaze tracking and scene understanding.

3 Requirements

3.1 Objective

In this work, the system aims to enhance autonomous navigation in human-populated indoor spaces by detecting and avoiding human obstacles in real-time while planning socially-aware paths around them. The primary objectives are to:

- Ensure safe, collision-free navigation in dynamic human environments
- Maintain efficient robot movement while adhering to social navigation norms
- Integrate real-time sensor fusion for improved dynamic obstacle representation

3.2 Functional Requirements

The system should meet the following functional requirements:

- **Real-time Obstacle Detection:** Utilize a combination of LiDAR and RGB-D camera data to recognize both static and dynamic obstacles.
- **Predictive Path Planning:** Incorporate trajectory prediction models to anticipate human movements and adjust navigation paths accordingly.
- **Adaptive Costmap Integration:** Dynamically update the local costmap to reflect detected obstacles and social navigation constraints.
- **Safety Compliance:** Prioritize zero collisions with human obstacles under normal operating conditions.

3.3 Operational Environment

The system was intended for deployment in structured indoor environments such as offices, university campuses, and retirement homes. These settings feature a mix of static and dynamic obstacles, including furniture and moving agents. Development was conducted in a pre-mapped hallway and room where human agents intentionally interfered with the robot's path to simulate real-world unpredictability, and add changes to the system to improve it. Environmental constraints include spaces for the robot to maneuver and lighting that provides visibility as it is camera-dependent.

3.3.1 Hardware Requirements

The experimental platform (Shown in Fig. 5) consists of the following core components:

- **Mobile Base:**
  - Model: Vstone MegaRover Ver3.0
  - Drive Type: Omnidirectional wheels
  - Payload Capacity: 40kg maximum
  - Platform: Custom acrylic mounting structure
- **Perception Sensors:**
  - 3D LiDAR: RoboSense RS-LiDAR-16
    - Channels: 16-beam
    - Field of View: 360° horizontal
    - Maximum Range: 150m
    - Output: 3D point cloud data
  - RGB-D Camera: Intel RealSense D435
    - Depth Range: 0.2m - 10m
    - Color Resolution: 1920×1080 @ 30fps
    - Output: Synchronized depth and RGB streams
  - Microphone
- **Computing Unit:**
  - Processor: x86_64 architecture
  - OS: Ubuntu 20.04 LTS

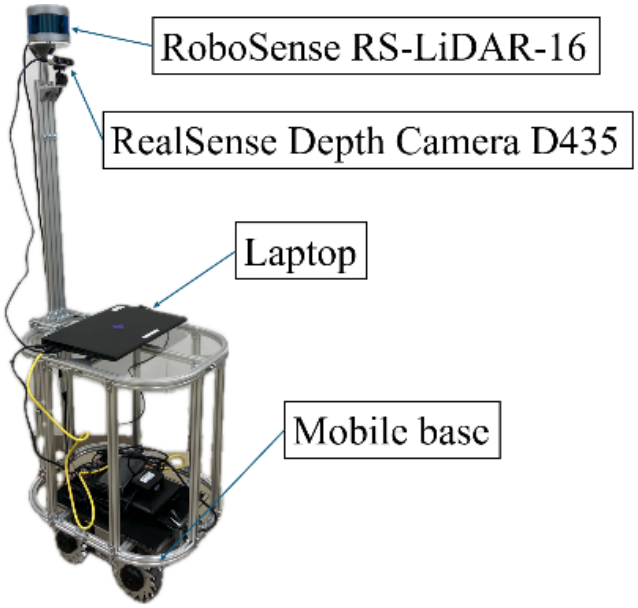


Figure 5. *MegaRover Ver3.0 mobile platform equipped with RS-LiDAR-16 and RealSense D435 sensors*

3.3.2 Software Requirements

- **Core Framework:** The presented approach has been implemented within ROS1 Noetic version on an HP OMEN 16-k0033dx with a Linux Ubuntu 20.04 environment with the following key packages:
  - `Navigation Stack`: ROS Navigation Framework and Global/local planning integration.
  - `costmap2d`: Multi-layer costmap management.
  - `custom costmap layer`: Developed in this work to translate detection data to costmap format and integrate trajectory predictions.
- **Perception Stack:** `Gmapping`: 2D mapping with LiDAR.
- **Dependencies:** GPU version of PyTorch, the NVIDIA CUDA Toolkit along with a Python 3 environment.

4 Methodology

This chapter details the technical approach used to implement camera-based detection and its integration into a dynamic obstacle avoidance system. The section outlines the software implementation, including human detection, cost map generation, autonomous navigation, trajectory prediction and the mechanisms for speech-to-intent processing. A subsequent section describes the system design and the experimental framework employed for baseline testing.

Use Case

The proposed system enables autonomous mobile robots to operate safely in human-populated environments through voice interaction and adaptive navigation. It is designed for environments such as hospitals, offices, or assistive settings, the robot accepts natural language commands and converts them into navigational goals using GPT-3.5 Turbo for intent extraction. A multimodal perception system—combining 3D LiDAR and an RGB-D camera detects and tracks nearby humans, dynamically adjusting the robot's path to maintain safe distances while accounting for predicted pedestrian motion. Figure 6 illustrates this use case scenario.

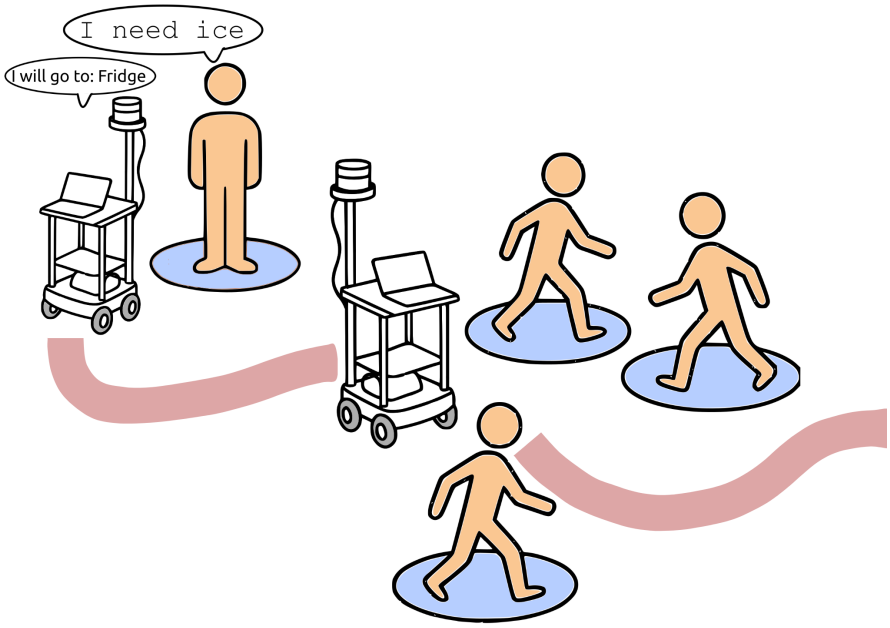


Figure 6. *Use Case Implementation*

4.1 Software Implementation

The system's operation begins when a user provides a navigation goal through voice commands. Upon detecting the activation word, the microphone captures the spoken command, which is then transcribed and interpreted into a navigational goal using speech-to-intent processing (Section 4.2). This goal serves as the input to the navigation stack.

For environment perception, the RoboSense RS-LiDAR-16 generates a 3D point cloud that is converted into a 2D laser scan, providing obstacle data for the base costmap layer. Simultaneously, the Intel RealSense D435 camera feeds RGB-D data to a YOLO-based detection pipeline that identifies humans in the robot's vicinity and estimates their 3D poses.

The system dynamically updates the costmap by combining:

- Obstacle data from the LiDAR sensor.
- Human pose from camera detection.
- Predicted human trajectories.

This integrated costmap enables the DWA local planner to compute safe, human-aware paths. The navigation execution module then guides the robot while continuously adapting to environmental changes. Figure 7 illustrates this complete workflow from voice input to motion execution.

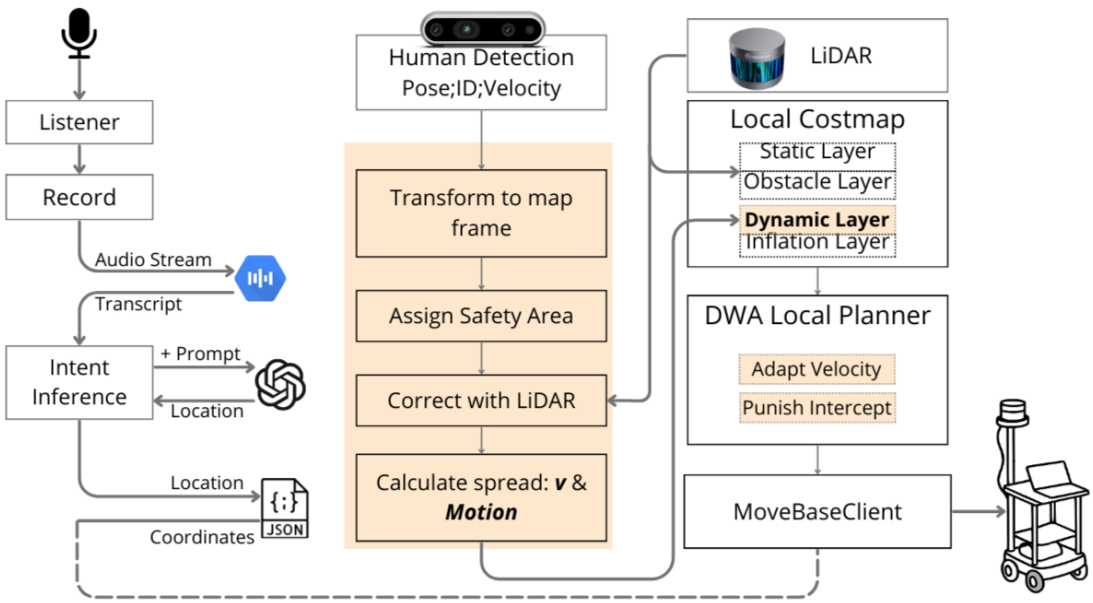


Figure 7. *Overview figure of the system*

4.2 Speech-to-Intent System Using GPT and Response Generation

To facilitate seamless human-robot interaction in populated environments, the robot is equipped with a speech-to-intent system that enables users to specify goal positions via voice commands. A microphone captures spoken input to ensure optimal audio reception.

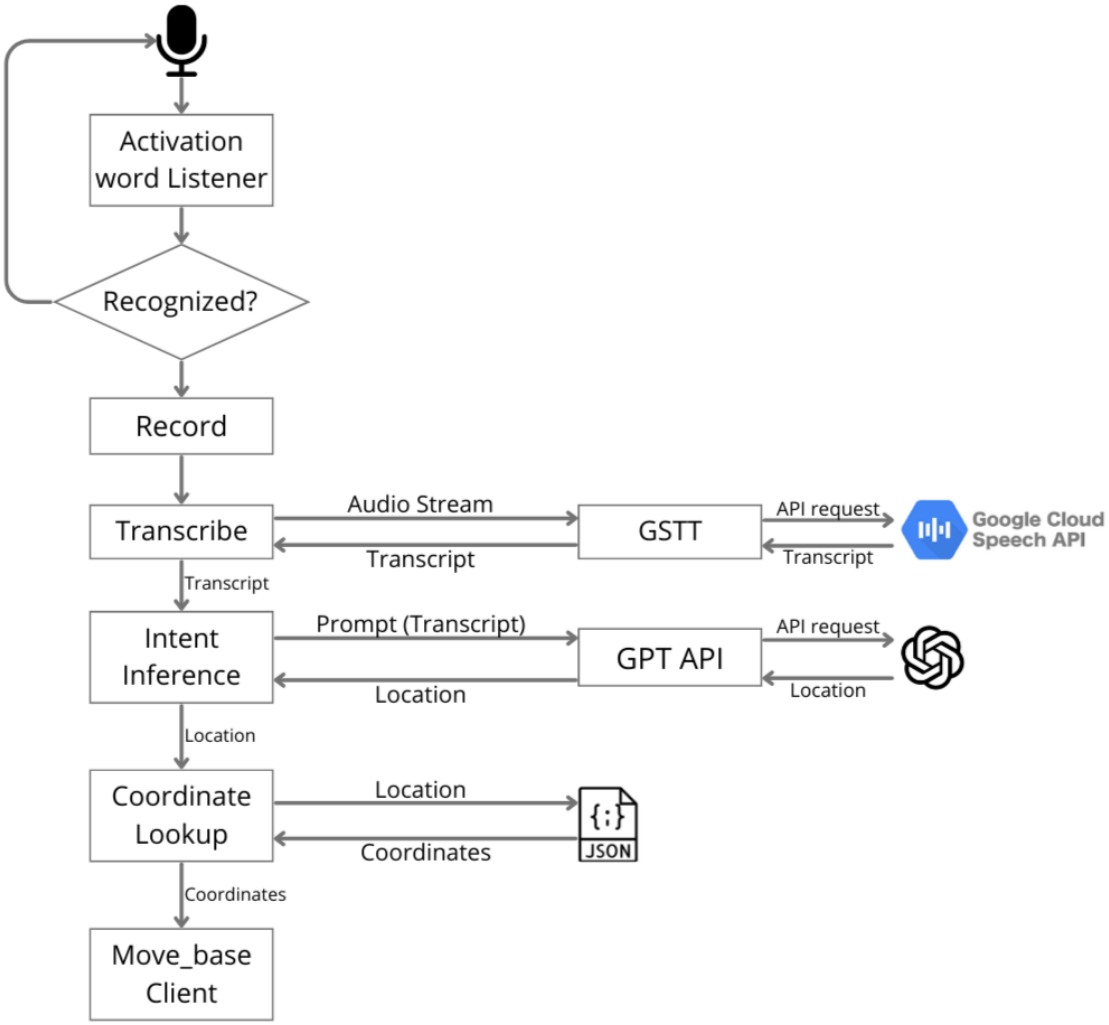


Figure 8. *Speech to intent pipeline*

The system operates in several stages:

1. **Speech Recognition:** Upon detecting the activation keyword *"start"*, the robot initiates a 10-second listening window to capture the user's command. Speech input is transcribed into text using `Google's Speech-to-Text (STT)` service, which supports up to 3 languages in its free version (In this case, English (`en-US`),

Spanish (`es-ES`), and Japanese (`ja-JP`)). The `recognize_speech()` function iterates through these languages until a valid transcription is obtained. If no speech is detected, the system logs a timeout error.

2. **Intent Inference via GPT-3.5 Turbo:** The transcribed text is processed by `OpenAI's GPT-3.5 Turbo model` using a structured prompt to enforce deterministic outputs. The system prompt is defined as:

   "You are an assistant that identifies locations based on user intent. The possible locations are List of location. Respond only with the location name, without additional text."

   The infer_location_intent() function parses the API response, enforcing deterministic outputs through regex validation. Successful inferences return canonical location names; failures trigger re-prompting via TTS.

3. **Goal Confirmation and Navigation:** The inferred location is mapped to preconfigured coordinates stored in a JSON file. The robot audibly confirms the destination using *Google's Text-to-Speech (TTS*) service, stating: *"I will go to: [GPT Response]"*. These coordinates are then dispatched to the navigation stack.

Multilingual support

The system automatically handles three languages (English, Spanish, Japanese) without pre-selection required, the language of choice can be modified accordingly. The `recognize_speech` function sequentially attempts transcription in each language until successful:

```
text = recognize_speech(languages=["es-ES", "en-US", "ja-JP"])
```

This method uses multilingual natural language and applies robust intent parsing, ensuring reliable operation. The modular design allows for future expansion to additional possible locations or commands minimal modifications.

Error Handling

If the speech-to-intent process fails to provide a goal location, the system emits an audible reprompt to restart the process. Persistent failures trigger a fallback to manual keyboard input. Additionally, once a goal location is confirmed and sent to the navigation stack, the robot will not accept additional voice commands until the current navigation task is completed. This prevents conflicting instructions during execution and ensures task consistency. Users must wait for the robot to reach its destination or manually interrupt the operation before issuing new commands.

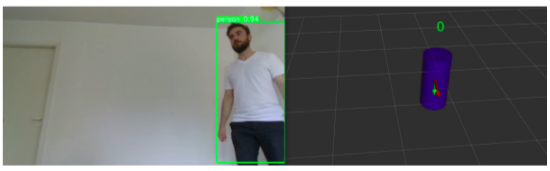
System Latency

Empirical measurements indicate a cumulative end-to-end intent resolution time of approximately 10 seconds. Further latency breakdowns are described in Section 4.6. Additionally, discussion of appropriate latency values for navigation, planning and execution is detailed in Section 6.

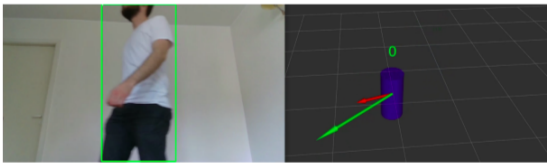
4.3 Human Detection Pipeline

The human detection system is incorporated from Joul's ROS-based 3D bounding box detection pipeline [28], [42], which provides real-time human localization using an RGB-D camera (See Fig. 9). The pipeline consists of three core components

1. **YOLOv5 Instance Segmentation**: A modified YOLOv5 model detects humans in RGB images and generates segmentation masks.
2. **3D Bounding Box Estimation**: Depth data from the RGB-D sensor is combined with segmentation masks to calculate 3D bounding boxes in the camera frame.
3. **Kalman Filter Tracking**: A multi-object tracker maintains temporal consistency using kinematic predictions.

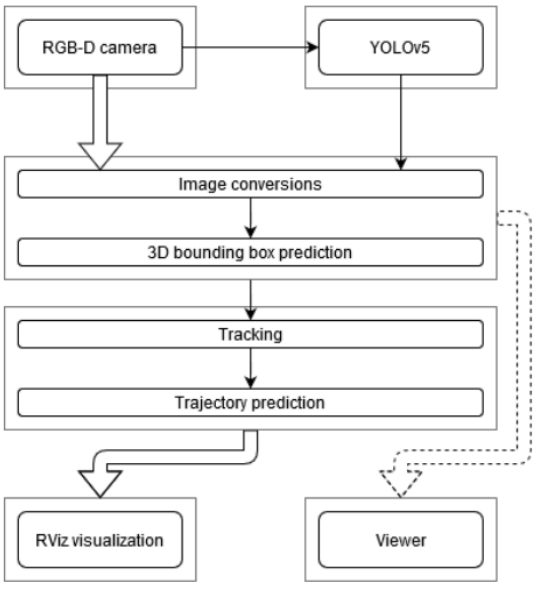


(a) Scene 1 starting position, object moving slightly



(b) Object moving

(a) *Visualization of 3D markers in RViz*



(b) *Detection pipeline overview*

Figure 9. *Human Detection Pipeline [28]*

The pipeline outputs `visualization_msgs/MarkerArray` messages containing:

- 3D bounding box vertices in camera coordinates
- Tracked human IDs with persistence across frames
- Velocity estimates derived from Kalman filter predictions

4.4 Costmap Layers and modifications

To address the challenge of integrating dynamic human poses—which are highly variable and sensitive to processing delays this work introduces a custom costmap layer. The approach developed in this area is divided in three steps:

1. **Object Transformation** – Starting from the camera sensor, humans are identified and converted to costmap representation.
2. **Scan Matching** – The transformed human pose are checked against the LiDAR scans and corrected.
3. **Cost assignment** – Assigns costs around each moving person in the local costmap according to a 2D asymmetric Gaussian shape with variances proportional to the obstacle velocity and oriented in its moving direction.

4.4.1 Object Transformation

The system processes real-time `visualization_msgs/MarkerArray` messages from the human detection pipeline, converting 3D bounding boxes into costmap-compatible obstacle data through a three-stage coordinate transformation. First, for each detection marker containing vertices $\{\mathbf{v}_1, \mathbf{v}_2\}$, the planar centroid is computed in the camera frame as:

$$\mathbf{c}_{\text{cam}} = \left[\frac{v_1^x + v_2^x}{2}, \frac{v_1^y + v_2^y}{2}, 0 \right]^T \tag{1}$$

This position is then transformed to the target navigation frame (`/map`) using the ROS TF2 library:

$$\mathbf{c}_{\text{map}} = \mathbf{T}_{\text{map}}^{\text{cam}} \mathbf{c}_{\text{cam}} \tag{2}$$

where $\mathbf{T}_{\text{map}}^{\text{cam}}$ is the homogeneous transformation matrix. Each detection is finally encoded as a `Person` message. The people_msgs/Person message format:

```
geometry_msgs/Point position # c_map coordinates
float64 reliability # Fixed at 1.0
float64 radius # 0.5m obstacle radius
string id # Unique number
```

All detected persons are aggregated into a `People` message and published to `/detected_people`, to serve as input for costmap inflation.

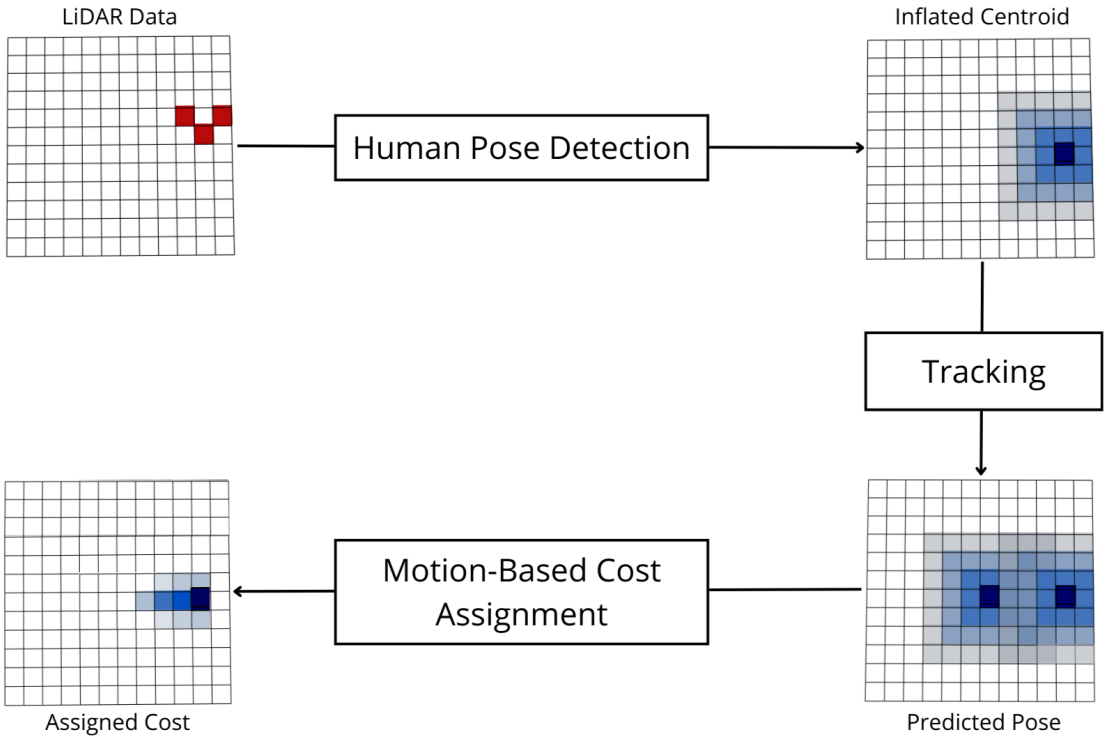


Figure 10. *Costmap layers with applied methods*

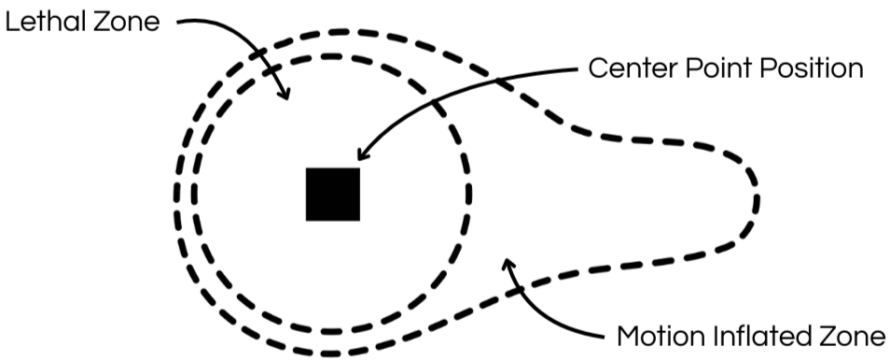


Figure 11. *Asymmetric Gaussian Shaping*

4.4.2 Scan Matching

During development it was noted that there could arise a situation in which the camera detections (therefore the position in map frame) and the LiDAR scans are not aligned. Under normal conditions these should be present in the same space, representing the human standing there. To eliminate this issue, the layer also checks their distance and aligns the position based on a LiDAR ground truth. This is achieved adding a correction step in the costmap layer. For each detected person, check if there are LiDAR points near the camera-estimated position and cluster them. If there are enough points,and these are not within a radius near the detected position, shift the detected position slightly toward the center of that LiDAR cluster.

4.4.3 Cost Assignment

To model dynamic human obstacles for safe and socially-aware navigation, the assigned costs to the local costmap uses both a fixed safety zone and a velocity-dependent Gaussian spread. The key idea is that faster-moving humans should create more extended cost regions in the direction of motion, allowing the robot to plan accordingly.

To ensure safety and efficient planning, two cost zones are defined:

- A fixed-radius **lethal zone** r_{lethal}, marking the area closest to the person as a non-traversable obstacle.
- A **directionally biased Gaussian zone**, inflated in the direction of motion based on velocity.

Fixed Radius First, a hard safety zone is established around the person's position, represented by a `geometry_msgs/Point`. Any grid cell within the fixed radius r_{lethal} is marked as a lethal obstacle, ensuring the robot maintains a safe distance regardless of the person's movement.

Gaussian Inflation Beyond the lethal zone, the cost is computed using a 2D Gaussian, but only within a limited region around the person defined by the inflation ranges. The Gaussian spread is applied only if the person is moving (Seen in Fig.12). Let $\mathbf{v}$ denote the speed. The longitudinal spread in the motion direction is given by:

$$\sigma_{\parallel} = \alpha \cdot \|\mathbf{v}\| \tag{3}$$

where α is a constant that control the spread of the Gaussian. A higher velocity leads to a larger $\sigma_{\parallel}$, resulting in an elongated inflation region in the direction of movement. This gives the robot more room to plan around faster-moving people in the direction of motion. However, after empirical testing if the velocity estimations were incorrect, the

costmap would be full with overshooting of spread. For that a σ_{max} was defined as the maximum allowed spread.

We define the Gaussian spread in the motion direction as:

$$\sigma(v) = \begin{cases} 0 & \text{if } \|\mathbf{v}\| = 0 \\ \min(\sigma_{\text{max}}, \alpha \cdot \|\mathbf{v}\|) & \text{if } \|\mathbf{v}\| > 0 \end{cases}$$

The parameters used in the final version after empirical testing are:

Table 1. *Key parameters for the dynamic human costmap layer. These can be adjusted via dynamic reconfigure according to testing.*

Parameter	Value	Description
r_{obstacle}	0.5	Lethal radius
α	1.0	Velocity scaling factor
σ_{max}	1.5	Maximum Gaussian spread

Once the Gaussian distribution is computed, it is applied to nearby grid cells in the local costmap. The cost assigned to each cell depends on how far it is from the center of the obstacle and its direction relative to the human's motion. The result is a costmap assignment in which the direction of movement shapes the cost and allows for a better path planning by the local planner. The cost assignment uses the following logic:

- If the cell is within the personal radius around the person, it is marked as the personal zone and assigned a lethal cost.
- If the cell is in front of the person (in the direction they are moving), it is assigned a cost based on a Gaussian function. The cost decreases as the cell gets further from the person in that direction. The faster the person moves, the further the cost spreads.
- If the cell is behind or lateral to the person's direction of motion, the cost is given by either the lethal zone or the inflation from the gaussian center.

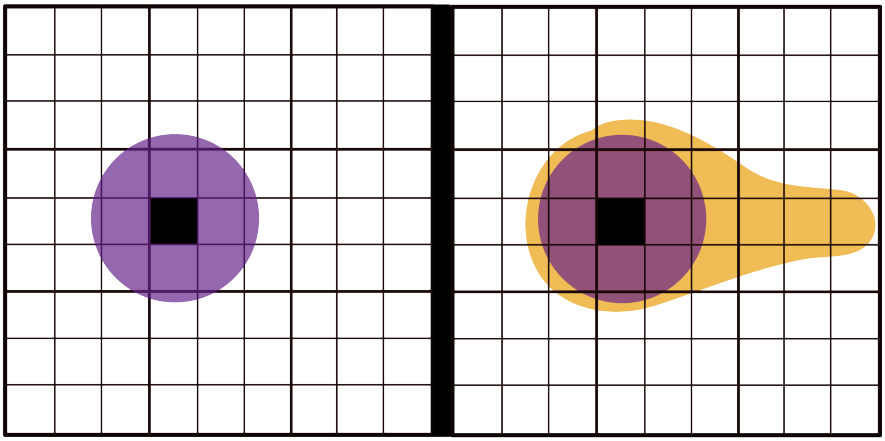


Figure 12. *ROS Costmap representation of human standing still (left) and moving towards the right direction (right)*

4.5 DWA Planner Modifications

The path planning process builds upon the 2D occupancy grid generated by the mapping algorithm using distance data from the LiDAR. This grid map forms the basis for computing obstacle-aware paths between goals, which are defined through speech-based intent parsing as described in 4.2. The ROS Navigation stack is used to configure both the global and local planners, where the global planner determines a feasible long-range route, and the local planner refines it dynamically based on real-time sensor updates [18].

As a result of the human representation design, human-aware components are introduced in the planning stack: direct avoidance, a velocity-sensitive obstacle inflation mechanism and velocity adaptation. These components are incorporated into both global and local costmaps for perceived social navigation and safety in human spaces.

1. **Direct Avoidance**: Each human is modeled as a distribution centered at their detected position, with a cost gradient towards their direction of motion. This influences the global path to naturally detour around humans field of view and to avoid direct motion in front of a person, preserving social norms. To further enforce this, a penalty was added to any trajectory heading straight toward a person's current position, making it less likely to be chosen as the path.

2. **Velocity-Obstacle Cost Assignment**: As faster moving humans represent a longer spread across the costmap, the planner can avoid intercept trajectories as it checks the velocity space and picks the highest scoring [43].

3. **Velocity Adaptation**: For each velocity sample, first evaluate the cost under the planned path. Then scale the robot's maximum forward speed (max_vel_x)

inversely with this cost, so in regions near high cost areas (human obstacles)—the robot automatically reduces its speed.

Overall, the local planner integrates these components not just to avoid collisions, but to further improve the results of the human dynamic layer and produce socially intelligent navigation strategies.

4.6 Latency

This section evaluates the latency introduced by each major component in the system, encompassing speech-based intent inference, visual perception, robot navigation, and actuation response.

Speech-to-Intent Pipeline

Empirical measurements indicate an average transcription latency of 1.7 seconds following the listening period, and a mean GPT-3.5 Turbo inference latency of 8.2 seconds. Together, this results in a cumulative end-to-end intent resolution time of approximately 10 seconds.

- **Keyword Detection:** 0.3s (audio buffer processing)
- **Speech-to-Text (Google STT):** 1.7s ± 0.4s (excluding the listening window)
- **GPT-3.5 Turbo Inference:** 8.2s ± 1.1s (including API round-trip)
- Subtotal: 10.2s ± 1.5s

Human Detection Pipeline

The computational tasks involved in the visual pipeline include 3D bounding box prediction, object tracking, and trajectory prediction. In the final implementation of the thesis the Human Detection nodes are executed on a GPU version of PyTorch and the NVIDIA CUDA Toolkit for faster computation. Even in scenarios with five detected objects, the average combined computation time for these modules remains below 10 milliseconds. This corresponds to a theoretical throughput of 100–300 frames per second [28].

When integrated with YOLOv5, which operates at an average rate of approximately 30 frames per second, the total computation time for the full visual pipeline (detection, tracking, and prediction) remains below 40 milliseconds per frame. These results demonstrate that visual processing is efficient and does not pose a computational bottleneck for real-time operation.

Actuation Latency

Actuation latency comprises delays in motor command processing and the robot's physical response. According to the specifications of the MegaRover Ver3.0 mobile platform [44]:

- **Motor Command Processing:** 5 ms (based on onboard MCU specifications)
- **Physical Response Delay:** 30–50 ms (mechanical response time for wheel initiation)

Although exact values can vary slightly with robot load and terrain, for the intended purpose these delays are minor compared to higher-level planning and perception latencies.

4.7 Resource Distribution

Real-time robotic navigation involving visual perception and dynamic obstacle avoidance presents significant computational demands. Initial development testing was conducted using a single laptop running all system components, including sensor drivers, the speech to action pipeline, the human detection pipeline (YOLO-based), the ROS navigation stack, and auxiliary nodes. However, this monolithic setup quickly revealed performance bottlenecks. Initial usage on a single laptop showed:

- **GPU Utilization:** YOLO inference consistently pushed GPU usage above 90%, leading to frame drops and delayed detections.
- **CPU Utilization:** Simultaneous ROS callbacks, LiDAR processing, costmap updates, and motion planning elevated CPU usage beyond 80%, frequently causing latency spikes exceeding 300 ms.

These issues had a direct negative impact on the robot's performance. Delayed or dropped detections led to outdated obstacle information being fed into the planner, increasing the inefficient or unsafe navigation decisions, specially in dynamic scenarios with crossing pedestrians. To mitigate these limitations and ensure timely data processing, a distributed architecture was adopted using a two-laptop setup connected over a wired network. This configuration enabled both computers to operate while sharing selected topics.

Network Topology and Role Assignment:

- **Laptop 1 (192.168.1.10)**: Perception Node: Designated for handling sensor input and computing-intensive tasks.

  - **Active Nodes:** LiDAR driver, Camera driver, Human Detection Pipeline.
  - **Primary Output:** Human position markers published on `/visualization_marker_array`.

- **Laptop 2 (192.168.1.20)**: Navigation Node: Focused on decision-making and control tasks.
  - **Active Nodes:** `move_base` node, Global Planner, Local Planner, Dynamic Obstacle Layer, Speech-to-Intent Interface.

Inter-machine communication used ROS topic sharing over TCP/IP. Only essential topics (such as detected human positions, LiDAR scans) were shared to reduce bandwidth consumption and latency (Fig. 13).

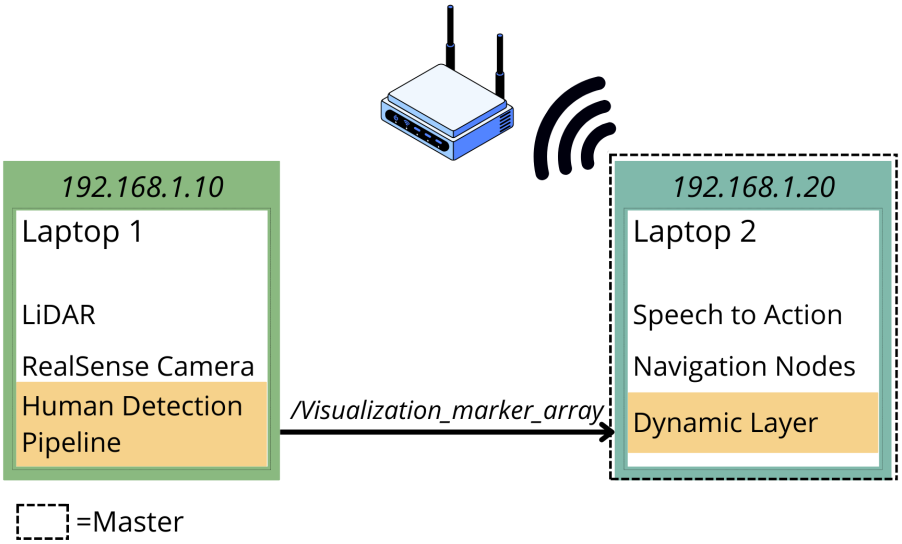


Figure 13. *Task Distribution Overview*

After deployment of the distributed configuration, system performance improved such that YOLO inference ran consistently at full frame rate, eliminating dropped detections. ROS callback latency dropped below 50 ms, even during high sensor throughput. The distribution resulted in real-time cost assignment and tracking of the human obstacles in the costmap, which was the main critical failure previously.

5 Results

5.1 Experimental Setup

Experiments were carried out in a 10-meter straight corridor (Shown in Fig. 14)containing static obstacles such as walls, columns, and trashcans, as well as up to three dynamic obstacles in the form of moving human participants. The primary objective was to evaluate navigation performance under varying levels of dynamic complexity; therefore, the speech interface was not employed. Instead, navigation goals were predefined and issued directly to the robot.

Each trial followed a three-phase process:

1. **Initialization:** The robot performed a self-check, localized itself within a preloaded global map, and initialized both global and local costmaps. Concurrently, the human detection module actively added any detected individuals to the local costmap as dynamic obstacles.

2. **Planning & Execution:**

   - The testing script dispatched a goal to the `move_base` node.
   - The global planner generated a path, while the local Dynamic Window Approach (DWA) planner refined it using real-time sensor inputs.
   - LiDAR and camera data updated the costmaps at frequencies of 10 Hz and 5 Hz, respectively.
   - The robot navigated autonomously, avoiding both static and dynamic obstacles.

3. **Completion:** Upon goal arrival, navigation metrics were logged automatically. Manual annotations were added when necessary before shutting down the nodes.

To increase the difficulty of the trials and simulate realistic dynamic interactions, human participants were instructed to deliberately obstruct the robot's path by walking across or standing in its trajectory. Importantly, participants were unaware to whether the system under evaluation was the baseline or the proposed planner, to reduce bias in their behavior. However, due to the inherent variability in human behavior, each trial introduced stochastic elements, particularly in the timing and trajectory of the human motion, which contributes to natural variation across runs.

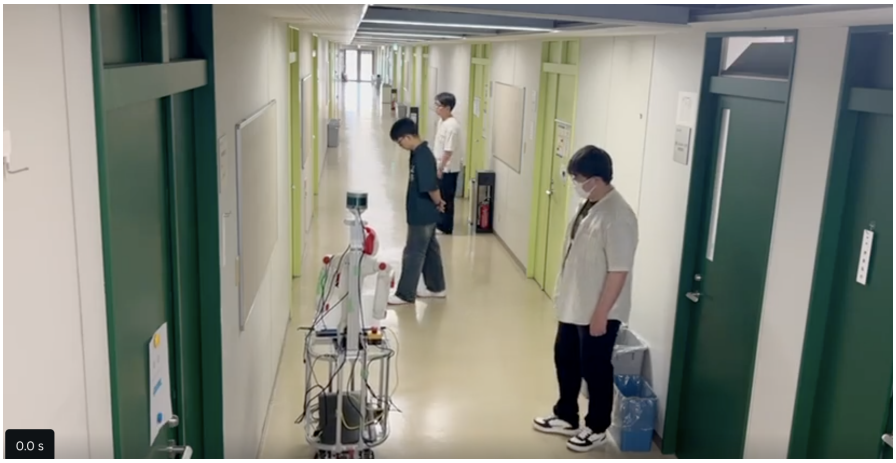


Figure 14. *Testing area with 3 human participants in a long hallway*

5.2 Evaluation Metrics

To evaluate improvements from the proposed implementation and compare performance, the baseline system—using a standard LiDAR-only DWA planner—was tested against the proposed system across ten trials per scenario. The following metrics were recorded:

- **Success Rate:** The percentage of trials in which the robot reached its goal without human intervention. Success was confirmed through:
  - The `move_base/GoalStatus` topic indicating successful completion.
  - Manual validation of the robot's final pose.
- **Collision Detection:** A collision event was defined as any instance where the distance between the robot's footprint and a human participant fell below 0.1 m. These events were detected using costmap-based proximity calculations and corroborated by participant feedback.
- **Time-to-Goal:** Total duration from the moment the goal was sent to the time the robot arrived at the target location. Timing was measured using timestamp logs captured in the evaluation script.
- **Motion Smoothness (Jerk):** Jerk, the derivative of acceleration, serves as an indicator of trajectory smoothness and responsiveness. Higher jerk values often reflect frequent or abrupt changes in direction, which can suggest inefficient or reactive replanning, especially in the presence of dynamic obstacles [45]. Acceleration was derived from `/odom` twist messages sampled at 100 Hz, excluding the initial 2 s to remove startup transients.

- **Recovery Behaviors:** The number of emergency stop or re-planning events triggered, recorded from the `move_base/recovery_behavior` topic.

5.3 Experiment Results

5.3.1 Success Rate and Failures

Across all three scenarios, the proposed planner delivers a consistent improvement success rate compared to the baseline. While the baseline degrades as crowd complexity increases, the proposed method does not drop as drastically as the number of human obstacles increase. This demonstrates that predictive, dynamic re-routing maintains high task completion even under sudden occlusions and multi-person crossings. As an overview, before even considering the remaining metrics the % of successful runs show the overall improvement.

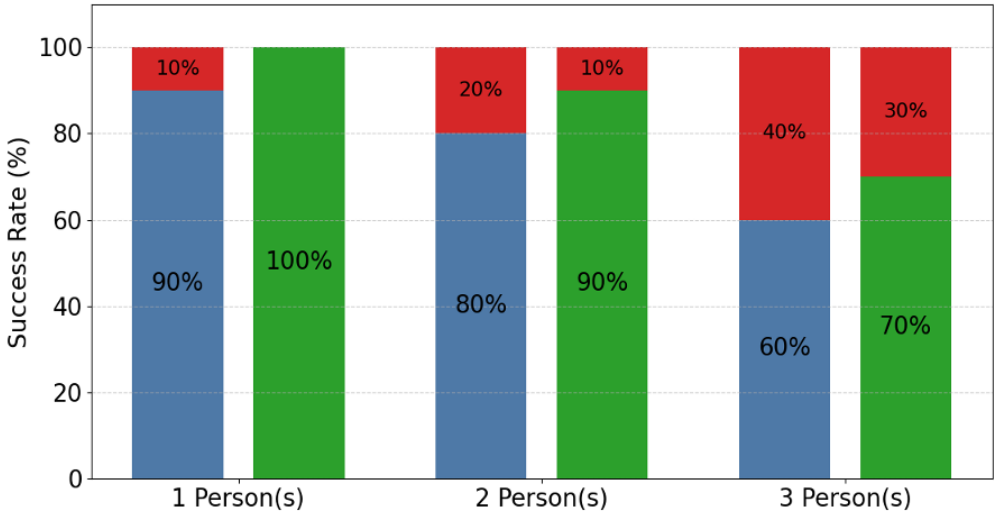


Figure 15. *Success Rate across experiments*

5.3.2 Collisions

Collisions directly measure the planner's reactive limitations. The baseline method results in increased collisions per trial as crowd density grows, reflecting its reliance on instantaneous LiDAR scans without foresight. In contrast, the proposed method holds collisions rarely. This is also due to the decision to label the human obstacle pose as a lethal obstacle and distributing the cost, highlighting the safety-first approach that was taken. In Experiment 2's staggered crossings, for instance, the robot using the baseline often misjudges the second person's entry and must swerve abruptly—triggering

a collision (Trajectory Shown in Fig.17). By modeling each human's projected path, the proposed method initiates smooth detours at a safe distance.

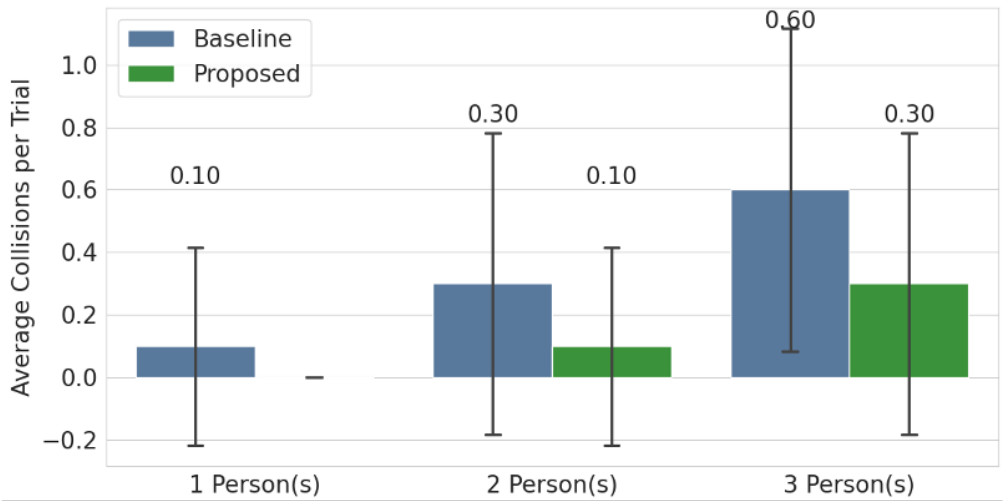


Figure 16. *Collision Rate across experiments*

(a) *ROS Standard* **(b)** *Proposed Planner*

Figure 17. *Trajectory results with different methods when avoiding a moving obstacle appearing from behind*

5.3.3 Time to Goal

Navigation time encapsulates both efficiency and the cost of reactive stops. The baseline's mean time grows with added people, revealing frequent stops and back-and-forth replanning. The proposed planner consistently shaves off time due to the predictive planning. In Experiment 3, the extra average 7.8s saving equates to almost 15% speedup,

despite the planner deliberately slowing when passing an obstacle. This speed gain arises because the proposed method allows the planner to identify "windows" in human movement and creates the path optimally, rather than stopping to re-scan each time an occlusion appears.

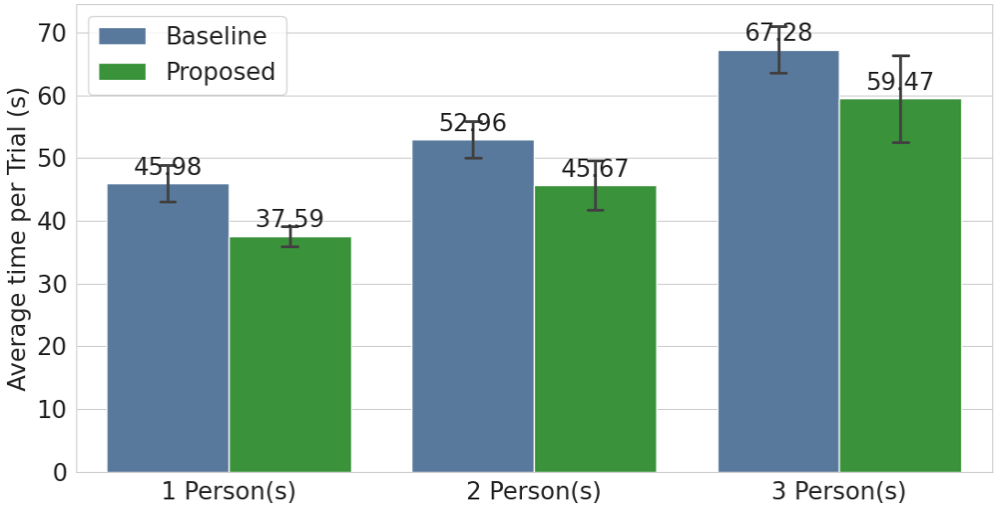


Figure 18. *Time per Trial across experiments*

5.3.4 Trajectory Smoothness (Jerk)

High jerk values indicate abrupt velocity changes, which translate to jerkiness in real robots and potential discomfort in human-robot interaction [46] [45]. The baseline's jerk averages 2.63 m/s³ across the scenarios, driven by sudden stop motions and rapid spinning to recover from deadlocks. By forecasting pedestrian motion and keeping track of the human obstacles position the robot can improve the average jerk to 2.3 m/s³—a reduction of up to 10%. The smoother profiles are evident when encountering a new person behind, rather than suddenly breaking the robot can smoothly avoid the area and re-plan (As seen in Fig.17)

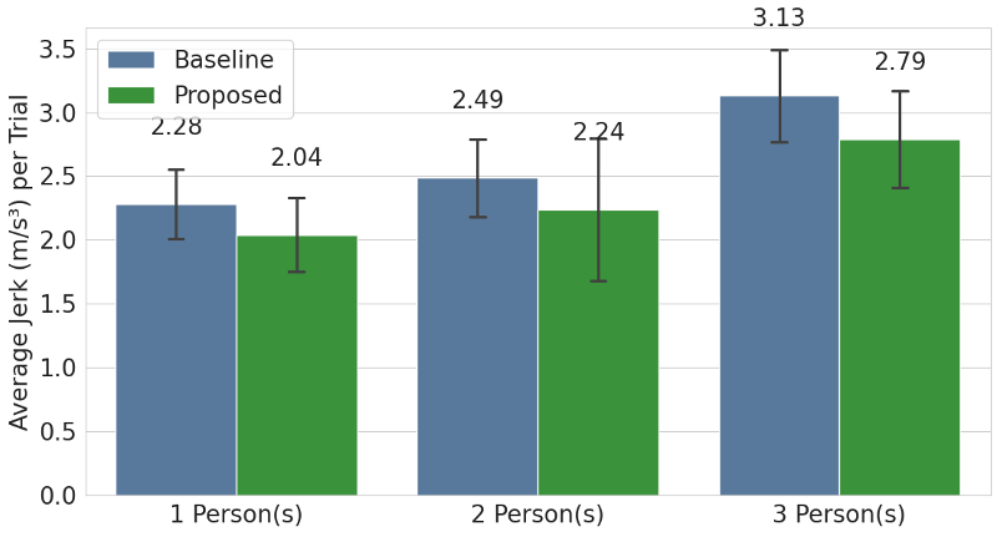


Figure 19. *Trajectory Smoothness (Jerk) across experiments*

5.3.5 Recovery Behaviors

Recovery actions quantify how often the robot "gets stuck" and is unable to create a plan without collisions. The baseline spikes from average recoveries per trial as crowd complexity increases, since unexpected occlusions cause it to spin in an attempt to understand clear obstacles. Our planner's recovery count remains near low, as a result of preempting and bypass tight spots altogether.

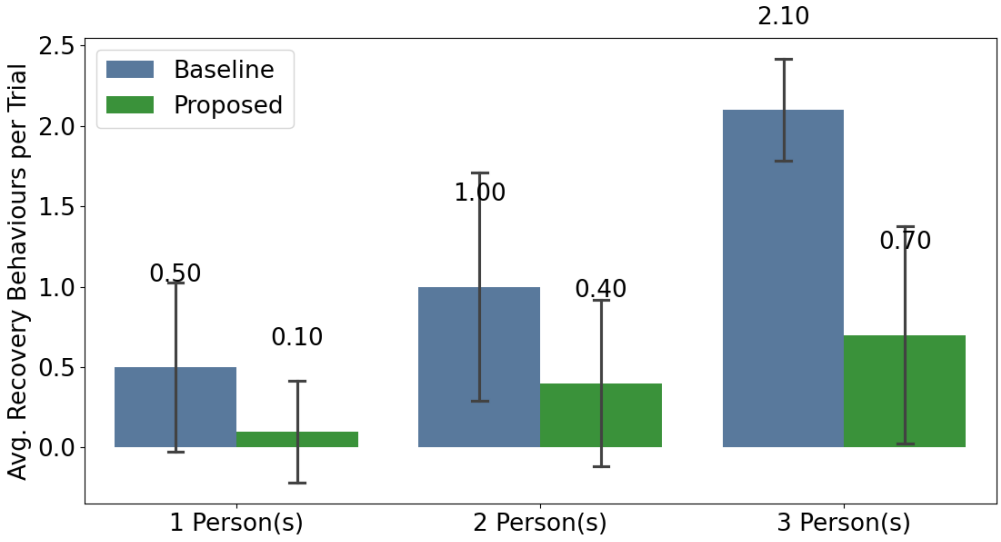


Figure 20. *Recovery Behaviours (Spin in Place) across experiments*

5.4 Experiment Breakdowns

Experiment 1: One Person Obstructing Path In this scenario, a single human crosses the corridor at different intervals. Table 2 presents the mean results for each method and the improvement. The Baseline method occasionally collides or fails when the human obstructs the path suddenly, incurring extra time and high jerk during repeated re-planning. In the 1-person experiment the participant was quickly moving to purposely block the path of the robot, so the main point of reduction is due to better planning with a long spread due to a fast velocity obstacle.

Table 2. *Experiment Performance Comparison: 1 Person*

Metric	Baseline Avg	Proposed Avg	Improvement $\pm$ SD	
Success Rate (%)	90.00	100.00	10.00	
Collisions	0.10	0.00	-0.10	±0.32
Time (s)	45.98	37.59	-8.40	±3.35
Jerk (m/s^3)	2.28	2.04	-0.24	±0.40
Recovery Behaviours	0.50	0.10	-0.40	±0.61

Experiment 2: Two People Staggered Crossing This experiment introduced increased complexity with two participants crossing at separate times. Table 3 summarizes the results. The baseline's success rate dropped further due to late-stage obstructions from the second human, leading to increased collisions and abrupt recovery actions. The proposed planner maintained smooth trajectories by anticipating motion patterns and proactively rerouting.

Table 3. *Experiment Performance Comparison: 2 People*

Metric	Baseline Avg	Proposed Avg	Improvement $\pm$ SD	
Success Rate (%)	80.00	90.00	10.00	
Collisions	0.30	0.10	-0.20	±0.58
Time (s)	52.96	45.67	-7.29	±4.94
Jerk (m/s^3)	2.69	2.54	-0.15	±0.48
Recovery Behaviours	1.00	0.40	-0.60	±0.88

Experiment 3: Three People This scenario represents the most challenging condition, involving three human participants, one of whom remained stationary to obstruct the corridor while the others move across the robot's path. As shown in Table 4, the baseline planner frequently failed in this setting, with collisions occurring regularly. A common

failure mode was the robot encountering a third person unexpectedly after passing a partially occluded individual—highlighting the limitations of relying solely on current sensor readings from laser scans without future motion prediction. These surprise encounters often triggered abrupt reactive maneuvers, resulting in high jerk values and up to four recovery behaviors per trial.

In contrast, the proposed method successfully completed most trials despite the increased crowd density. Although the time-to-goal was slightly longer due to more cautious planning, the robot maintained smoother trajectories, evidenced by moderate jerk values and minimal recovery actions.

Table 4. *Experiment Performance Comparison: 3 People*

Metric	**Baseline Avg**	**Proposed Avg**	**Improvement ± SD**	
Success Rate (%)	60.00	70.00	10.00	
Collisions	0.60	0.30	−0.30	±0.71
Time (s)	67.28	59.47	−7.81	±7.88
Jerk (m/s^3)	3.13	2.79	−0.34	±0.52
Recovery Behaviours	2.10	0.70	−1.40	±0.75

6 Discussion

In this section the results and limitations of the work are discussed. To improve the system and reduce limitations, some ideas are proposed for further development.

This thesis developed a method to incorporate human pose information from a depth camera into the costmap layers of the ROS navigation stack, aiming to enhance dynamic obstacle avoidance in environments shared with humans. Additionally, a pipeline was implemented to process spoken commands, infer user intent, and navigate to corresponding goal locations using the proposed planning framework. The entire methodology is implemented as modular ROS packages, available on GitHub [47][48]. The discussion is organized by system components.

Speech-to-intent The system makes use of large language models (LLMs) for parsing user intent from spoken commands. Specifically, GPT-3.5 Turbo is used for intent recognition. While the use of commercial APIs raises potential concerns about cost, this does not represent an issue since the usage remains minimal, in this context—each request consumes approximately 74 tokens, resulting in negligible cost (around $0.0001 USD per request at current pricing).

The pipeline is designed to be modular and easily extensible. New available target goals or command types can be added simply by updating the `coordinates.json` file and modifying the system prompt. This ensures flexibility and scalability with minimal changes to the codebase.

Human Robot Interaction As discussed in Section 4.6 the system's speech-to-intent latency exceeds the commonly cited 2-second threshold for natural turn-taking in conversation,however, prior research in human-robot interaction demonstrates that user tolerance can extend to higher latencies in task-oriented interactions, particularly when system complexity justifies the delay [49]. Perceived latency can be mitigated through the use of verbal fillers that signal active processing. This principle is applied and in the case of this project is seen as a terminal message *"Processing your request..."*.

Human Detection Observations False positives in human detection were occasionally observed due to the model interpreting human-like shapes—such as a long coat hanging on a rack or a humanoid robot in the lab—as actual humans. These objects were temporarily added to the costmap as dynamic obstacles. However, once the detector corrected its prediction, they were subsequently cleared from the map. While these occurrences were not systematically analyzed in this work, they were noted during testing and development.

Experiment Size A notable limitation in evaluation stems from the small number of test participants. A broader user base would have provided more robust data and potentially revealed edge cases or user-specific challenges. Additionally, a wider range of evaluation metrics could have been employed to better assess system performance. However, due to time constraints—particularly because development occurred in Japan and needed to be concluded before returning to Estonia—such extensions were not feasible during this phase.

6.1 Limitations

Several limitations arise from the sensors of the system. As the system relies primarily on a camera for human detection, making its performance sensitive to lighting conditions. In low-light scenarios, the camera may fail to detect humans, though the fallback on LiDAR allows continued obstacle detection and navigation. Additionally, real-world environments such as homes or public spaces can have significant background noise, which may degrade the performance of the speech recognizer and result in incorrect or failed intent inference.

Another limitation is the system's reliance on cloud-based LLMs for natural language understanding. In areas with limited or unreliable internet access, this dependency can

hinder the functionality. Future versions of the system would benefit from incorporating local or hybrid models to improve robustness in offline or constrained environments.

7 Conclusion

This thesis focused on enabling mobile robots to navigate safely and socially among humans in indoor environments by proposing a method for dynamic obstacle avoidance as a costmap layer for ROS Navigation. It is shown how the problem can be tackled within the ROS framework by developing both a human-aware costmap layer and modifying a planner integrated into ROS Navigation. In the current state of the work, the custom costmap layer transforms real-time 3D human pose detections into velocity- and direction-sensitive Gaussian inflations around each person, which is used by the local planner to provide a safer navigation plan.

For demonstrating the implementation in human spaces, a Speech to intent parser was developed using a large language model (GPT 3.5 Turbo) for parsing user intent from spoken commands. This resulted in the robot being able to interpret spoken voice vague commands and determine the goal location based on the extracted intent.

The developed system was compared against a baseline LiDAR-only standard-DWA planner. Experimental validation in corridor trials with up to three participants demonstrated that the proposed method achieves a 10% higher success rate, reduces collisions by up to 50%, and decreases traversal time by up to 15% compared to the baseline. However there are limitations to the proposed method and future work should address remaining limitations and challenges:

- Incorporating human-human dynamics to further develop social awareness, such as taking head pose into account and number of human agents in the area.
- Implementing more sophisticated detection and prediction models. To get more accurate human pose information and remove false detections.
- Extending evaluation to larger, more complex environments with varied obstacle patterns and more participants to quantify performance in true real-world settings.
- Incorporating user comfort metrics and dynamic human–robot interaction studies to refine proxemic modeling.

By addressing these areas, the methodology can be made more robust, responsive, and adaptable to a wide range of socially-aware robotic applications.

8 Acknowledgments

I would like to thank

My supervisors Bin Zhang and Karl Kruusamäe for their guidance and support during the thesis development as well as my Master Degree.

My parents for supporting me always even from a distance and enabling me to study far from home.

The University of Tartu for supporting my studies and having opportunities that allowed me to choose the path I desired.

Kanagawa University & Zhang Lab. for welcoming me as one of their own.

References

[1] Fabian Hart and Ostap Okhrin. "Enhanced method for reinforcement learning based dynamic obstacle avoidance by assessment of collision risk". In: *Neurocomputing* 568 (Feb. 2024), p. 127097. ISSN: 09252312. DOI: 10.1016/j.neucom.2023.127097. URL: https://linkinghub.elsevier.com/retrieve/pii/S0925231223012201 (visited on 04/04/2025).

[2] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. 2nd ed. Intelligent robotics and autonomous agents. OCLC: ocn649700153. Cambridge, Mass: MIT Press, 2011. 453 pp. ISBN: 978-0-262-01535-6.

[3] Wikipedia. *Starship Technologies*. URL: https://en.wikipedia.org/wiki/Starship_Technologies.

[4] Suhaila Mubarrat Jaffna and Prithila Bhowmik. *Kiva Robotics System: Revolutionizing Automation and Expanding Healthcare Applications*. Nov. 28, 2024. DOI: 10.33774/coe-2024-r9tzn. URL: https://www.cambridge.org/engage/coe/article-details/673e10095a82cea2faf0b55c (visited on 04/15/2025).

[5] Ezgi Uzel Aydınocak. "Robotics Systems and Healthcare Logistics". In: *Health 4.0 and Medical Supply Chain*. Ed. by İsmail İyigün and Ömer Faruk Görçün. Series Title: Accounting, Finance, Sustainability, Governance & Fraud: Theory and Application. Singapore: Springer Nature Singapore, 2023, pp. 79–96. ISBN: 978-981-99-1817-1 978-981-99-1818-8. DOI: 10.1007/978-981-99-1818-8_7. URL: https://link.springer.com/10.1007/978-981-99-1818-8_7 (visited on 04/15/2025).

[6] Robert Groot. "Autonomous Exploration and Navigation with the Pepper robot". Master Thesis. Department of Information and Computer Sciences: Utrecht University, Oct. 2018.

[7] Parkview Health. *Aethon TUG*. URL: https://www.parkview.com/blog/a-new-fleet-of-tugs-goes-to-work.

[8] Gonzalo Fuentes. *Pepper Robot*. URL: https://www.livemint.com/mint-lounge/business-of-life/pepper-robot-remains-alive-and-well-says-japan-s-softbank-111625041581806.html.

[9] Abeysekara Nadeesha Dhananji and Tharaga Sharmilan. "Autonomous Mobile Robot Navigation and Obstacle Avoidance: A Comprehensive Review". In: *European Modern Studies Journal* 7.6 (Feb. 5, 2024), pp. 260–267. ISSN: 2522-9400. DOI: 10.59573/emsj.7(6).2023.25. URL: https://www.journal-ems.com/index.php/emsj/article/view/982 (visited on 04/11/2025).

[10] Anbalagan Loganathan and Nur Syazreen Ahmad. “A systematic review on recent advances in autonomous mobile robot navigation”. In: *Engineering Science and Technology, an International Journal* 40 (Apr. 2023), p. 101343. ISSN: 22150986. DOI: 10.1016/j.jestch.2023.101343. URL: https://linkinghub.elsevier.com/retrieve/pii/S2215098623000204 (visited on 04/11/2025).

[11] Silvia Guillén-Ruiz et al. “Evolution of Socially-Aware Robot Navigation”. In: *Electronics* 12.7 (Mar. 27, 2023), p. 1570. ISSN: 2079-9292. DOI: 10.3390/electronics12071570. URL: https://www.mdpi.com/2079-9292/12/7/1570 (visited on 04/11/2025).

[12] Khawla Almazrouei, Ibrahim Kamel, and Tamer Rabie. “Dynamic Obstacle Avoidance and Path Planning through Reinforcement Learning”. In: *Applied Sciences* 13.14 (July 13, 2023), p. 8174. ISSN: 2076-3417. DOI: 10.3390/app13148174. URL: https://www.mdpi.com/2076-3417/13/14/8174 (visited on 04/04/2025).

[13] Weiwei Zhao et al. “A Study of the Global Topological Map Construction Algorithm Based on Grid Map Representation for Multirobot”. In: *IEEE Transactions on Automation Science and Engineering* 20.4 (Oct. 2023), pp. 2822–2835. ISSN: 1545-5955, 1558-3783. DOI: 10.1109/TASE.2022.3198801. URL: https://ieeexplore.ieee.org/document/9861387/ (visited on 04/04/2025).

[14] Boston Dynamics. *Strech Robot*. URL: https://bostondynamics.com/products/stretch/.

[15] Albashir A. Youssef et al. “Brief Survey on Industry 4.0 Warehouse Management Systems”. In: *International Review on Modelling and Simulations (IREMOS)* 15.5 (Oct. 31, 2022), p. 340. ISSN: 2533-1701, 1974-9821. DOI: 10.15866/iremos.v15i5.22923. URL: https://www.praiseworthyprize.org/jsm/index.php?journal=iremos&page=article&op=view&path[]=27121 (visited on 04/17/2025).

[16] Morgan Quigley et al. “ROS: an open-source Robot Operating System”. In: *ICRA workshop on open source software*. Vol. 3. Issue: 3.2. Kobe, Japan, 2009, p. 5.

[17] Li Zhi and Mei Xuesong. “Navigation and Control System of Mobile Robot Based on ROS”. In: *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). Chongqing: IEEE, Oct. 2018, pp. 368–372. ISBN: 978-1-5386-4509-3. DOI: 10.1109/IAEAC.2018.8577901. URL: https://ieeexplore.ieee.org/document/8577901/ (visited on 04/10/2025).

[18] Pangcheng David Cen Cheng et al. “Dynamic Path Planning of a mobile robot adopting a costmap layer approach in ROS2”. In: *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA). Stuttgart, Germany: IEEE, Sept. 6, 2022, pp. 1–8. ISBN: 978-1-6654-9996-5. DOI: 10.1109/ETFA52439.2022.9921458. URL: https://ieeexplore.ieee.org/document/9921458/ (visited on 04/08/2025).

[19] *ROS Navigation*. URL: https://wiki.ros.org/navigation.

[20] Zishu Chai and Zongyuan Zhang. “Mobile Robot Path Planning in 2D space: A survey”. In: *2022 International Symposium on Control Engineering and Robotics (ISCER)*. 2022 International Symposium on Control Engineering and Robotics (ISCER). Changsha, China: IEEE, Feb. 2022, pp. 47–57. ISBN: 978-1-6654-8478-7. DOI: 10.1109/ISCER55570.2022.00015. URL: https://ieeexplore.ieee.org/document/9894511/ (visited on 04/12/2025).

[21] Saeid Nahavandi et al. “A Comprehensive Review on Autonomous Navigation”. In: *ACM Computing Surveys* (Mar. 31, 2025), p. 3727642. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3727642. URL: https://dl.acm.org/doi/10.1145/3727642 (visited on 04/11/2025).

[22] Xiaomao Zhou, Yanbin Gao, and Lianwu Guan. “Towards Goal-Directed Navigation Through Combining Learning Based Global and Local Planners”. In: *Sensors* 19.1 (Jan. 5, 2019), p. 176. ISSN: 1424-8220. DOI: 10.3390/s19010176. URL: https://www.mdpi.com/1424-8220/19/1/176 (visited on 04/09/2025).

[23] Houman Masnavi. “MULTI-ROBOT MOTION PLANNING FOR SHARED PAYLOAD TRANSPORTATION”. PhD thesis. Tartu: University of Tartu, 2020.

[24] Shaoshuai Shi et al. *PV-RCNN: The Top-Performing LiDAR-only Solutions for 3D Detection / 3D Tracking / Domain Adaptation of Waymo Open Dataset Challenges*. Version Number: 1. 2020. DOI: 10.48550/ARXIV.2008.12599. URL: https://arxiv.org/abs/2008.12599 (visited on 04/16/2025).

[25] Flavio B.P. Malavazi et al. “LiDAR-only based navigation algorithm for an autonomous agricultural robot”. In: *Computers and Electronics in Agriculture* 154 (Nov. 2018), pp. 71–79. ISSN: 01681699. DOI: 10.1016/j.compag.2018.08.034. URL: https://linkinghub.elsevier.com/retrieve/pii/S0168169918302679 (visited on 04/13/2025).

[26] Kezhi Wang et al. “LiDAR-Only Ground Vehicle Navigation System in Park Environment”. In: *World Electric Vehicle Journal* 13.11 (Oct. 27, 2022), p. 201. ISSN: 2032-6653. DOI: 10.3390/wevj13110201. URL: https://www.mdpi.com/2032-6653/13/11/201 (visited on 04/13/2025).

[27] *Computer Vision at Tesla*. Think Autonomous. URL: https://www.thinkautonomous.ai/blog/computer-vision-at-tesla/.

[28] Juri Joul. "3D bounding box detection of moving objects for robot navigation in dynamic environments". PhD thesis. Tartu: University of Tartu, 2021. 40 pp.

[29] Hind Messbah, Mohamed Emharraf, and Mohammed Saber. "Robot Indoor Navigation: Comparative Analysis of LiDAR 2D and Visual SLAM". In: *IAES International Journal of Robotics and Automation (IJRA)* 13.1 (Mar. 1, 2024), p. 41. ISSN: 2722-2586, 2089-4856. DOI: 10.11591/ijra.v13i1.pp41-49. URL: https://ijra.iaescore.com/index.php/IJRA/article/view/20645 (visited on 04/04/2025).

[30] Zhiyu Pan et al. *LiCamPose: Combining Multi-View LiDAR and RGB Cameras for Robust Single-frame 3D Human Pose Estimation*. July 16, 2024. DOI: 10.48550/arXiv.2312.06409. arXiv: 2312.06409[cs]. URL: http://arxiv.org/abs/2312.06409 (visited on 04/04/2025).

[31] Jianhong Wu. "Research on Obstacle Avoidance Control of Intelligent Robots Based on Multi-Sensor Fusion". In: *Highlights in Science, Engineering and Technology* 114 (Oct. 31, 2024), pp. 24–30. ISSN: 2791-0210. DOI: 10.54097/r4qytk13. URL: https://drpress.org/ojs/index.php/HSET/article/view/25552 (visited on 04/04/2025).

[32] *robotics-upo/nav2_social_costmap_plugin*. original-date: 2023-01-15T12:27:31Z. May 16, 2025. URL: https://github.com/robotics-upo/nav2_social_costmap_plugin (visited on 05/18/2025).

[33] Barış Bilen. *bilenbaris/Social_Robot_Nav*. original-date: 2022-06-01T07:07:05Z. Dec. 10, 2024. URL: https://github.com/bilenbaris/Social_Robot_Nav (visited on 05/18/2025).

[34] Gerardo Pérez et al. "Social Elastic Band with Prediction and Anticipation: Enhancing Real-Time Path Trajectory Optimization for Socially Aware Robot Navigation". In: *International Journal of Social Robotics* (Apr. 29, 2024). ISSN: 1875-4791, 1875-4805. DOI: 10.1007/s12369-024-01135-z. URL: https://link.springer.com/10.1007/s12369-024-01135-z (visited on 05/18/2025).

[35] Ricarte De Sousa Ribeiro and Plinio Moreno. "Socially Reactive Navigation Models for Mobile Robots in Dynamic Environments". In: *Robot 2023: Sixth Iberian Robotics Conference*. Ed. by Lino Marques et al. Vol. 976. Series Title: Lecture Notes in Networks and Systems. Cham: Springer Nature Switzerland, 2024, pp. 222–234. ISBN: 978-3-031-58675-0 978-3-031-58676-7. DOI: 10.1007/978-3-031-58676-7_18. URL: https://link.springer.com/10.1007/978-3-031-58676-7_18 (visited on 05/18/2025).

[36] Matej Dobrevski and Danijel Skocaj. “Adaptive Dynamic Window Approach for Local Navigation”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Las Vegas, NV, USA: IEEE, Oct. 24, 2020, pp. 6930–6936. ISBN: 978-1-7281-6212-6. DOI: 10.1109/IROS45743.2020.9340927. URL: https://ieeexplore.ieee.org/document/9340927/ (visited on 05/18/2025).

[37] Ha Quang Thinh Ngo et al. “Develop the socially human-aware navigation system using dynamic window approach and optimize cost function for autonomous medical robot”. In: *Advances in Mechanical Engineering* 12.12 (Dec. 2020), p. 1687814020979430. ISSN: 1687-8132, 1687-8140. DOI: 10.1177/1687814020979430. URL: https://journals.sagepub.com/doi/10.1177/1687814020979430 (visited on 05/18/2025).

[38] Sumin Kang et al. “Social Type-Aware Navigation Framework for Mobile Robots in Human-Shared Environments”. In: *Sensors* 24.15 (July 26, 2024), p. 4862. ISSN: 1424-8220. DOI: 10.3390/s24154862. URL: https://www.mdpi.com/1424-8220/24/15/4862 (visited on 05/18/2025).

[39] Yang Zheng, Yongkang Liu, and John H.L. Hansen. “Intent detection and semantic parsing for navigation dialogue language processing”. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). Yokohama: IEEE, Oct. 2017, pp. 1–6. ISBN: 978-1-5386-1526-3. DOI: 10.1109/ITSC.2017.8317620. URL: http://ieeexplore.ieee.org/document/8317620/ (visited on 04/17/2025).

[40] Wenlong Huang et al. *Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents*. Version Number: 2. 2022. DOI: 10.48550/ARXIV.2201.07207. URL: https://arxiv.org/abs/2201.07207 (visited on 04/21/2025).

[41] Haicheng Liao et al. “GPT-4 enhanced multimodal grounding for autonomous driving: Leveraging cross-modal attention with large language models”. In: *Communications in Transportation Research* 4 (Dec. 2024), p. 100116. ISSN: 27724247. DOI: 10.1016/j.commtr.2023.100116. URL: https://linkinghub.elsevier.com/retrieve/pii/S2772424723000276 (visited on 04/21/2025).

[42] Juri Joul. *Jyrijoul/ros_3d_bb*. original-date: 2021-05-20T19:44:10Z. Dec. 17, 2024. URL: https://github.com/Jyrijoul/ros_3d_bb (visited on 04/24/2025).

[43] Kaiyu Zheng. *ROS Navigation Tuning Guide*. Version Number: 2. 2017. DOI: 10.48550/ARXIV.1706.09068. URL: https://arxiv.org/abs/1706.09068 (visited on 05/11/2025).

[44] *MecanumRover Ver.3.0 Manual*. URL: https://www.vstone.co.jp/products/wheelrobot/ver.3.0_mecanum.html.

[45] Jee-Eun Lee et al. "On the Performance of Jerk-Constrained Time-Optimal Trajectory Planning for Industrial Manipulators". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024 IEEE International Conference on Robotics and Automation (ICRA). Yokohama, Japan: IEEE, May 13, 2024, pp. 9772–9778. ISBN: 979-8-3503-8457-4. DOI: 10.1109/ICRA57147.2024.10610437. URL: https://ieeexplore.ieee.org/document/10610437/ (visited on 05/08/2025).

[46] Huber, Markus, and Lenz. "Human preferences in industrial human-robot interactions". In: *Proceedings of the international workshop on cognition for technical systems*. 2008.

[47] *PaoAvalos/Speech_to_Intent: Speech to intent module of the MSc thesis work. From a microphone input, retrieve the coordinates for the intended goal location.* URL: https://github.com/PaoAvalos/Speech_to_Intent (visited on 05/20/2025).

[48] PaoAvalos. *PaoAvalos/DynamicObstacleLayer*. original-date: 2025-04-22T15:59:04Z. May 20, 2025. URL: https://github.com/PaoAvalos/DynamicObstacleLayer (visited on 05/20/2025).

[49] Hannah Pelikan and Emily Hofstetter. "Managing Delays in Human-Robot Interaction". In: *ACM Transactions on Computer-Human Interaction* 30.4 (Aug. 31, 2023), pp. 1–42. ISSN: 1073-0516, 1557-7325. DOI: 10.1145/3569890. URL: https://dl.acm.org/doi/10.1145/3569890 (visited on 04/23/2025).

Appendix

The implementations described in the thesis were developed as a whole but separated for reproducibility and clarity into different GitHub repositories.

- The speech to navigation goal module: https://github.com/PaoAvalos/Speech_to_Intent
- The proposed costmap layer: https://github.com/PaoAvalos/DynamicHumanLayer
- The modified DWA planner in ROS: https://github.com/PaoAvalos/SocialDWA

To implement in simulation or real-conditions the Human Detection Pipeline developed by Joul,J. is needed. Another detection pipeline could be used but it would be necessary to rename and re-structure [42]

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Paola Avalos Conchas**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Socially Aware Planning for Indoor Navigation,

   supervised by Bin Zhang, Ph.D. at Kanagawa University and Karl Kruusamäe, Ph.D. at University of Tartu.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive license does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Paola Avalos Conchas
20/05/2025