



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



DEPARTMENT OF INFORMATION ENGINEERING
MASTER DEGREE IN COMPUTER ENGINEERING

Active perception based on 2D LiDAR for social navigation in crowded environments

Master Candidate
Guglielmin Niccolò
Student ID 2052422

Supervisor
Prof. Bellotto Nicola
University of Padova

Co-Supervisor
Prof. Gloria Beraldo

Academic Year 2022/2023
Graduation Date 30/11/2023

*To all the people, places and
moments that experienced
my happiness, perseverance, and efforts*

To my family, to sincere friends

Thank you

Abstract

In the domain of social robotics, where robots dynamically interact with humans, the fusion of laser data with vision systems and Machine Learning (ML) techniques is crucial for navigation toward target people. Active perception not only reduces uncertainties but also enhances the sociability of robots by granting them a more profound comprehension of the social context in which they operate.

In this thesis new active perception methods for social navigation in crowded environments are proposed that are based on 2D Light Detection and Ranging (LiDAR) in order to be used in public space by respecting privacy and without requiring additional sensors. The proposed methods consider a policy that measures and exploits the level of uncertainty in the people detection to choose the most useful poses for the robot to confirm the detection or to deny a previous wrong detection. The other two methods concern learning this policy based on a Deep Neural Network (DNN) and learning via Reinforcement Learning (RL) the most appropriate robot's motions, to increase the certainty about the people in the environment.

Results about the policy behavior show that this approach is feasible to be used on a real robot than the other two approaches that were tested only in simulations. Furthermore, the experiments at the IAS-Lab on the Take It And Go (TIAGo)++ revealed that the robot can adapt its behavior to the working scenario, in which it can accomplish its tasks successfully when deployed in challenging situations, that is the goal of active perception.

Sommario

Nel campo della robotica sociale, dove i robot interagiscono dinamicamente con gli esseri umani, la fusione dei dati laser con i sistemi di visione e le tecniche ML è cruciale per la navigazione verso le persone target. La percezione attiva non solo riduce le incertezze ma migliora anche la socialità dei robot garantendo loro una comprensione più profonda del contesto sociale in cui operano.

In questa tesi vengono proposti nuovi metodi di percezione attiva per la navigazione sociale in ambienti affollati che si basano su 2D LiDAR per essere utilizzati nello spazio pubblico rispettando la privacy e senza richiedere sensori aggiuntivi. I metodi proposti considerano una politica che misura e sfrutta il livello di incertezza nel rilevamento delle persone per scegliere le pose più utili al robot per confermare il rilevamento o negare un precedente rilevamento errato. Gli altri due metodi riguardano l'apprendimento di questa politica basata su una DNN e l'apprendimento tramite RL dei movimenti più appropriati per il robot, per aumentare la certezza sulle persone nell'ambiente.

I risultati sul comportamento delle politiche mostrano che questo approccio può essere utilizzato su un robot reale rispetto agli altri due approcci che sono stati testati solo nelle simulazioni. Inoltre, gli esperimenti presso lo IAS-Lab sul TIAGo++ hanno rivelato che il robot può adattare il suo comportamento allo scenario di lavoro, in cui può svolgere i suoi compiti con successo se impiegato in situazioni difficili, quale è l'obiettivo della percezione attiva.

Contents

List of Tables	xi
List of Algorithms	xv
List of Acronyms	1
1 Introduction	1
1.1 Socially aware robot navigation	1
1.1.1 Problem Formulation	2
1.1.2 Social Conventions	4
1.1.3 Modeling Human Motion	6
1.1.4 Context-Aware Mapping	9
1.2 Core challenges	11
1.2.1 Planning Challenges	11
1.2.2 Behavioral Challenges	12
1.2.3 Evaluation challenges	13
1.2.4 Open Problems	14
1.3 Aim of the thesis	16
1.4 Structure of the thesis	16
2 State-of-art	19
2.1 Geometrical Approaches	19
2.1.1 RGB-D and Laser Data Fusion based Human Detection and Tracking	19
2.1.2 Dynamic Social Zone (DSZ) based Mobile Robot Navigation	24
2.1.3 Approaching Pose Prediction	29
2.2 Social Force Model (SFM)-based Approaches	33
2.2.1 Social Force Model for pedestrian dynamics	33
2.2.2 Waypoint-based path planner	36

2.3	Learning-based Approaches	41
2.3.1	Human Trajectory Prediction in Crowds	42
2.3.2	Genetic algorithm for learning to plan people-aware trajectories	45
2.3.3	Neuro-Symbolic Approach for Enhanced Human Motion Prediction	48
2.4	Active Perception	50
2.4.1	Classical Approaches	50
2.4.2	Active Perception	56
3	Tools and methods	59
3.1	Robot Operating System (ROS)	59
3.1.1	GAZEBO	60
3.1.2	ROS Visualization (RViz)	61
3.1.3	Distance Robust SPatial Attention and Auto-regressive Model (DR-SPAAM)	62
3.1.4	Pedestrian Simulator (PedSim)	63
3.1.5	OpenAI Reinforcement Learning (OpenAI RL)	64
3.2	Take It And Go (TIAGo++) robot	65
3.2.1	Versatility	65
3.2.2	Hardware architecture	66
3.2.3	Software architecture	67
3.2.4	Simulated TIAGo Base robot	69
4	Preprocessing: People detection and laser points classification	71
4.1	Obstacles interpolation method	71
4.2	Template Computation for detecting people	73
4.3	People Detection method	78
4.4	Intermediary test to verify the preprocessing approach	83
4.4.1	Crowded environment	83
4.4.2	No people environment	85
4.4.3	Simple office with people environment	85
4.5	Discussion of the intermediary test	88
5	Social Active Perception	91
5.1	Reinforcement Learning (RL): Introduction	93
5.2	Q-learning	95

5.3	Introduction to the proposed approaches	96
5.3.1	Intuition behind the proposed approach	96
5.4	Design and Implementation of the Proposed Approaches	97
5.4.1	Deterministic Policy	97
5.4.2	Learning from data	99
5.4.3	Reinforcement Learning	99
5.5	Clustering analysis for points grouping	101
5.5.1	Preliminary tests on clustering	102
5.5.2	First experimented configuration	103
5.5.3	Second experimented configuration	105
5.5.4	Third experimented configuration	107
5.5.5	Clusterings comparison conclusions	108
5.6	Mathematical formulation of the problem	109
5.6.1	Elements formalization	109
5.6.2	Design of the problem	113
5.6.3	Example of algorithm executions	118
5.6.4	Reward function	123
5.6.5	Example of reward function application	126
5.7	Learning from data	129
5.7.1	Neural Network (NN) for policy learning	129
5.7.2	Parameters for policy learning	130
5.7.3	Results of learned policy	130
5.8	RL Training	134
5.8.1	RL training elements	134
5.8.2	Deep Q-Network (DQN) for RL training	135
5.8.3	RL training parameters	136
5.8.4	Results of RL training	139
5.9	Simulated World	140
6	Experimental Results	145
6.1	Test Environment	146
6.2	Test Configurations	147
6.3	Evaluation metrics	149
6.4	Results	150
6.4.1	Results on only obstacles	150
6.4.2	Results on group vs. single target	151
6.4.3	Results on queue of targets	155

6.4.4	Results on occlusion to target	158
6.4.5	Results on occlusion to object	162
6.4.6	Results on double target occlusion with objects	164
6.4.7	Accuracy comparison	167
6.4.8	Correlation over data	168
6.5	Assessment Questionnaire	170
7	Conclusions	175
7.1	Discussions	175
7.2	Future works	176
	References	179
	Acknowledgments	189

List of Tables

1.1	Evaluation metrics for social navigation proposed in [59].	14
2.1	Comparison of w/out waypoint selection approaches on the characteristic of path properties taken from [48].	41
2.2	Performance comparison between the average relative gain across all datasets considered. The results' format refers to the 8/12 prediction time steps. ADE and FDE values are in meters. The results are taken from [61].	50
2.3	Performance comparison between the baseline architecture DARNN and the NeuroSyM approach on the JackRabbit dataset. The results' format refers to the 48/80 prediction time steps. RMSE and MAE values are in meters, and the best results are highlighted in bold (i.e. the lower error, the better). The results are taken from [61].	50
2.3	Analysis of the state-of-the-art approaches based on the following parameters: (a) Single or Multi pedestrian; (b) Pedestrians prediction; (c) Group awareness; (d) Crowds; (e) Link to code; (f) Evaluation metrics; (g) Simulation tests; (h) Real world tests. If a work takes into account single or multi pedestrians, then symbols M or S are respectively added. For the other features, a tick or a cross will be added according to the study of that feature in the work.	55
2.3	Collection of approaches where title and work define the analyzed paper. In the Gap column the limitations of such approaches are reported.	58
3.1	TIAGo++'s main specifications.	68

4.1	Table of indexes computed for both methods applied to the crowded environment.	84
4.2	Table of indexes computed for both methods applied to the simple office with people environment.	88
5.1	Table of metrics displayed in the histograms in Figure 5.9 and in Figure 5.10.	104
5.2	Table of metrics displayed in the histograms in Figure 5.12 and in Figure 5.13.	106
5.3	Table of metrics displayed in the histograms in Figure 5.15 and in Figure 5.16.	107
5.4	General state vector for RL training.	110
5.5	Table of indexes computed for all groups in the LIKELY clustering. The bold cluster is the one with maximum utility for the potential Go Likely action.	119
5.6	Table of indexes computed for all groups in the UNDEFINED clustering. The bold cluster is the one with maximum utility for the potential Go Undefined action.	119
5.7	Table of indexes computed for all potential actions, to decide which can have an higher utility. The bold action is the one that will be executed.	120
5.8	Table of indexes computed for all groups in the LIKELY clustering. The bold cluster is the one with maximum utility for the potential Go Likely action.	121
5.9	Table of indexes computed for all groups in the UNDEFINED clustering. The bold cluster is the one with maximum utility for the potential Go Undefined action.	121
5.10	Table of indexes computed for all potential actions, to decide which can have an higher utility. The bold action is the one that will be executed.	121
5.11	Table of indexes computed for all groups in the LIKELY clustering. The bold cluster is the one with maximum utility for the potential Go Likely action.	122
5.12	Table of indexes computed for all groups in the UNDEFINED clustering. The bold cluster is the one with maximum utility for the potential Go Undefined action.	122

5.13	Table of indexes computed for all potential actions, to decide which can have an higher utility. The bold action is the one that will be executed.	123
5.14	Sign analysis of the reward function components.	124
5.15	Gain in detection: analysis of the reward function components. . .	126
5.16	Loss in detection: analysis of the reward function components. . .	127
5.17	Parameters for the policy learning: the number of steps for a single epoch changes according to the number of data collected during simulated runs of the policy.	130
5.18	Parameters for the policy learning (<i>DANGER</i> scenario): 33.00 minutes, 58.61 seconds.	132
5.19	Parameters for the policy learning (<i>DISCOVER</i> scenario): 29.00 minutes, 31.47 seconds.	132
5.20	Parameters for the Deep Q-Network (DQN) network training. . .	138
5.21	Parameters for the Q-learning and the RL training.	138
5.22	Parameters for the task environment definition.	138
5.23	Parameters for the Social Active Perception task.	138
6.1	Summary table with analysis on correlation over people-robot distances and people detection confidences. The correlation indices are in bold. The linear regression coefficients are the angular coefficient m_{lr} and the intercept q_{lr} of the linear regression line. . . .	168
6.2	Table of metrics displayed in the histograms in Figure 5.12 and in Figure 5.13.	170

List of Algorithms

1	Kinodynamic RRT	22
2	Estimate Approaching Goal Pose	26
3	Waypoints (Social Subgoals) Selection	39
4	Genetic training	47
5	General circle interpolation <i>points_interpolation(P)</i>	73
6	Human Feet Detection Parameters <i>compute_parameters(P)</i>	75
7	Group Laser Data Points <i>group_data_points(P)</i>	76
8	Interpolate Group Points <i>compute_template_parameters(g)</i>	77
9	Laser data to Points <i>laser_data_to_points(scan_topic)</i>	79
10	Initialize new message <i>init_new_message(M)</i>	80
11	Update Message to Publish <i>update_message(msg, g_i)</i>	80
12	Classify groups <i>classify_groups(g, new_scan_msg, sab_scan_msg)</i>	81
13	People Detection and Classification <i>people_detection_classification(</i> <i>scan_topic, μ_r, σ_r)</i>	82
14	General Reinforcement learning training	94
15	Robot Working Policy	113
16	Learning from policy data runs	129
17	Robot RL training step	135

List of Acronyms

ADE Average Displacement Error

ARI Adjusted Rand Index

CAM Context-Aware Mapping

CND Conceptual Neighbourhood Diagram

CPU Central Processing Unit

CRF Conditional Random Field

DA Dual-stage Attention

DART Dynamic Animation and Robotics Toolkit

DBSCAN Density-Based Spatial Clustering of Applications with Noise

DL Deep Learning

DNN Deep Neural Network

DoF Degrees of Freedom

DQN Deep Q-Network

DR-SPAAM Distance Robust SPatial Attention and Auto-regressive Model

DSZ Dynamic Social Zone

EKF Extended Kalman Filter

EPS Extended Personal Space

FDE Final Displacement Error

FMI Fowlkes-Mallows index

FoV Field of View

FoVs Field of Views

FROG Friendly Robot Outdoor Guide

GA Genetic Algorithm

GB GigaByte

GHz GigaHertz

HM Human Motion

HRI Human-Robot Interaction

IoT Internet of Things

LiDAR Light Detection and Ranging

LRP Layerwise Relevance Propagation

LSTM Long Short-Term Memory

LTS Long-Term Support

MAE Mean Absolute Error

MARF Multiscale Adaptive-switch Random Forest

MATLAB Matrix Laboratory

ML Machine Learning

MLP Multi-Layer Perceptron

NMI Normalized Mutual Information

NN Neural Network

OCR Optical Character Recognition

ODE Open Dynamics Engine

OGRE Object-Oriented Graphics Rendering Engine

OpenCV Open Source Computer Vision Library

OPTICS Ordering Points To Identify Cluster Structure

PedSim Pedestrian Simulator

PCL Point Cloud Library

QTC Qualitative Trajectory Calculus

RAM Random Access Memory

ReLU Rectified Linear Unit

RFID Radio Frequency IDentification

RGB Red Green Blue

RGB-D Red Green Blue-Depth

RL Reinforcement Learning

RMSE Root Mean Square Error

RNN Recurrent Neural network

ROS Robot Operating System

RRT Rapidly-exploring Random Trees

RViz ROS Visualization

SARN Socially Aware Robot Navigation

SC Social Conventions

SFM Social Force Model

SGAN Social Generative Adversarial Network

SIS Social Interaction Space

SLAM Simultaneous Localization And Mapping

SPENCER Social situation-aware perception and action for cognitive robots

TIAGo Take It And Go

UAVs Unmanned Aerial Vehicles

URDF Unified Robot Description Format

XML Extensible Markup Language

Chapter 1

Introduction

In this chapter the introduction to the main general topic of Socially Aware Robot Navigation (SARN) is provided through a couple of definitions to formalize the topic, a general overview of the key aspects, and a specification of the core challenges in this field of research. Then, the motivation and the structure of this thesis are reported.

1.1 Socially aware robot navigation

Navigation is an essential skill for autonomous robots, and it becomes a challenging task in human-populated environments. Robots need to perform the tasks without disturbing the humans around them and ensure the comfort and safety of humans as well [80]. There are various factors influencing this behavior like social norms, the geometry of the environment, and surrounding people. To establish an effective socially aware robot navigation, the basic and fundamental components that must be acquired are Social Conventions (SC), Human Motion (HM) and Context-Aware Mapping (CAM).

By the fact that humans move from one place to another driven by some inner motivation towards a goal following a nonlinear pattern, one of the most desirable and complex task for a robot is to predict the accurate human trajectory in real-time. The difficulty increases because of human random behavior, surrounding people, social rules, and the environment. This is the reason why robots need to understand the context in the form of CAM to achieve a highly efficient SARN.

More precisely, human behavior, that is reflected by HM, can be considered to be random since human people can decide to change their own goals at any instant. Despite this, human motion is mostly influenced by both physical and

social constraints that can be found in the particular context and environment (Figure 1.1).



(a) ETH-Univ (b) ETH-Hotel (c) UCY-Zara (d) UCY-Students (e) UCY-Arx.

Figure 1.1: Different snapshots of scenarios with pedestrians taken from common datasets proposed in [5].

However, in social environment settings, a robot is preferred to be reactive enough to deal with these constraints, deliberating a suitable plan of motion execution. For example, some general SC can be in support of human order of motion, the respecting and the following of the queue, the motion to the left or to the right-hand side of the way (according to each country), to ask for permission (whenever required), etc.

Since tracking humans and obstacles is a tricky job for a robot in motion, in general is preferred to predict human trajectories to plan the robot navigation in advance, taking into account the most potential and probable human choices. For this reason, formalization of both problem and potential situations is needed.

1.1.1 Problem Formulation

[59] proposes both formulations about the concepts of navigation and social navigation that are reported in this subsection.

Definition of Navigation

Consider a planar workspace $C \subseteq \mathbb{R}^2$ and denote by $C_{obs} \in C$ the subspace of C that is occupied by static obstacles. Assume that an agent (an individual or a group) a is embedded in C , lying at a configuration $q \in Q \subseteq SE(2)^1$ and occupying a finite area $A(q) \subseteq C$. Starting from q , the agent intends to reach a goal $g \in Q$ while avoiding collisions with the static environment. For agent a , **Navigation** is the task of following a collision-free path $\tau : [0, 1] \rightarrow Q$ connecting q to g . To this end, agent a is solving a problem of the form:

¹SE(2) is the Special Euclidean group. It is a subgroup of the direct Euclidean isometries whose elements are called rigid motions or Euclidean motions. They comprise arbitrary combinations of translations and rotations, but not reflections.

$$\begin{aligned}
\tau &= \underset{\tau \in \mathcal{T}}{\operatorname{argmin}} && c(\tau) \\
\text{s.t.} &&& A(\tau(a)) \notin C_{obs}, t \in [0, 1] \\
&&& \tau(0) = q \\
&&& \tau(1) = g
\end{aligned}$$

where t is a normalized notion of time, \mathcal{T} is a space of paths, and $c : \mathcal{T} \rightarrow \mathbb{R}$ is a cost function describing additional path specifications (e.g., time to goal or path smoothness) and taking into account environmental properties (e.g., terrain traversability) [59].

Definition of Social Navigation

Consider a set of $n_t > 1$ agents a_i , $i \in N = \{1, \dots, n_t\}$, lying at configurations $q_i \in Q$ with volumes $A_i(q_i)$ at some time $t \geq 0$. Agents intend to reach their individual destinations g_i while avoiding collisions with the static environment and abiding by social norms (e.g., respecting the personal space of others). At planning time, agent i generates a path $\tau_i : [0, 1] \rightarrow Q$, extracted by solving **Social Navigation** as an optimization problem of the form:

$$\begin{aligned}
\tau_i &= \underset{\tau \in \mathcal{T}}{\operatorname{argmin}} && c_i(\tau_i) + \lambda_i c_i^s(\tau_i, \tilde{\tau}_{-i}) \\
\text{s.t.} &&& A_i(\tau_i(a)) \notin C_{obs}, \forall t \in [0, 1] \\
&&& A_i(\tau_i(a)) \cap A_j(\tilde{\tau}_j(a)) = \emptyset, \forall t \in [0, 1], j \neq i \\
&&& \tau(0) = q_i \\
&&& \tau(1) = g_i
\end{aligned}$$

where $c_i : \mathcal{T} \rightarrow \mathbb{R}$ is an individual cost corresponding to an individual path specifications; $c_i^s : \mathcal{T}^2 \rightarrow \mathbb{R}$, a cost² describing social considerations, such as personal space, taking into account predictions about the future behaviour of others $\tilde{\tau}_{-i} = (\tilde{\tau}_2, \dots, \tilde{\tau}_{n_t})$; and λ_i a weight. It is assumed that agents do not have access to the complete navigation parameters of others. Specifically, it is assumed that at planning time agent i is not aware of the navigation costs c_j , c_j^s , and weight λ_j , τ_j [59].

²This cost function formulation is very generic to capture the widest variety of possible behaviors. It spaces from social compliance to adversarial behavior.

1.1.2 Social Conventions

Social Behaviours

Human behaviors can be analyzed from different points of view like sociology and psychology. The crucial point is to connect the right social cues to the right social signals by humans (facial expressions, gestures, body posture, proximity, etc). This is because robot navigation must ensure social comfort³, which is very subjective and cannot be measured directly.

Proxemics

The term “Proxemics” refers to the study of maintaining spatial distances in various interpersonal and social spaces [80]. Figure 1.2 shows the four types of spaces that a socially aware robot should manage to avoid discomfort.

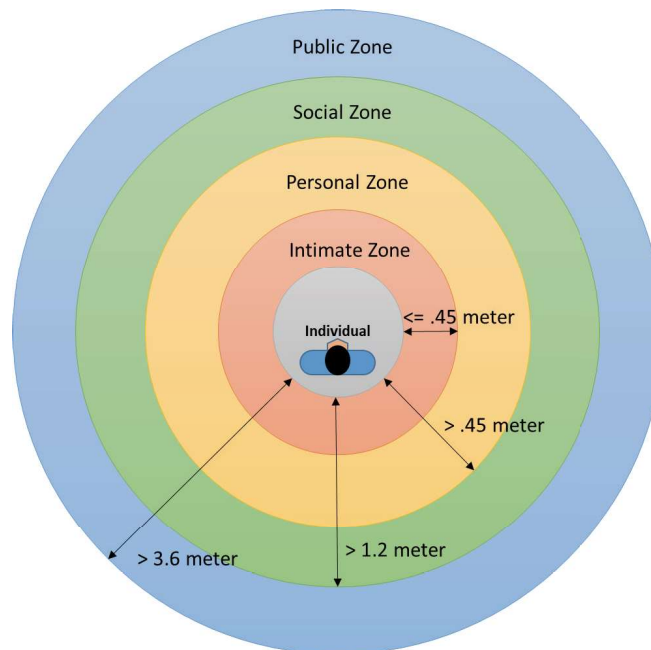


Figure 1.2: Classification of personal space proposed in [80].

About groups, the related space changes according to how two or more individuals are engaged in conversation [80]. Figure 1.3 shows the seven types of group spaces.

It must be taken into account that the space around any individual is situation-dependent and, in general, dynamic in time. Indeed, Proxemics is strictly related

³Social comfort can be defined as the absence of stress and irritation while interacting with robots [80].

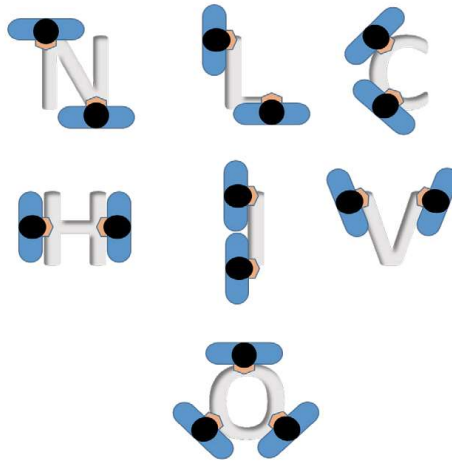


Figure 1.3: Different types of group space configurations proposed in [80].

to other factors like the space between individuals and objects, robot appearance, robot speed, and approach directions. Practically, some properties and behaviors are suggested and preferred since they are more effective [81]. Such behaviors can be related to robot speeds, head movements, and gaze direction [44].

Social Robot Capabilities

“Social Navigation” includes not only navigation itself. Since robots must establish natural interaction similar to humans, approaching directions towards humans are of fundamental importance. For this reason, a detailed knowledge of human representation and space can help plan an effective and as correct as possible SARN.

Social robots must be able to treat humans differently from objects while navigating in a predictable and easily understandable manner. The needed capabilities can be classified into four types (as shown in Figure 1.4):

1. *Avoiding collision*: it is established by Social Force Model (SFM)⁴ [49];
2. *Passing humans*: robots compute the cost for each selectable trajectory by considering parameters and choosing a left or right pass when encountering stationary humans [65];
3. *Following humans*: navigation is carried out in crowded scenarios by robots considering a target location and selecting a human leader to follow;

⁴An example of the Social Force Model applied to robotics can be found in [27].

4. *Moving along with humans*: motion is carried out in crowded environments while taking optimum robot speed and braking into consideration, accompanying a human in the motion to a goal position [73].

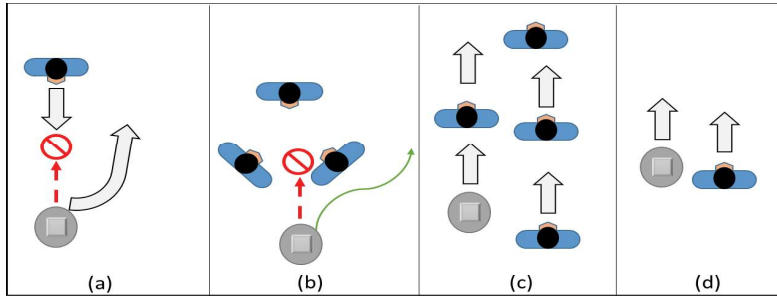


Figure 1.4: Classification of social robot capabilities proposed in [80]: (a) Avoiding collision. (b) Passing humans. (c) Following humans. (d) Moving along with humans.

1.1.3 Modeling Human Motion

Modeling human motion is a complex task. It aims at generating future data based on the observed, captured, and computed human motion state. This task can be divided into three types of approaches: Physics-Based Approaches, Pattern-Based Approaches and Planning-Based Approaches.

Physics-based approaches

These types of approaches use Newton's laws of motion to predict human trajectories (how humans move) and their possible interactions with the environment. According to the different modes of dynamics, they can be divided into the following types:

- *Single-model approaches*: position and velocity are enough to represent human motion state in the form of a kinematic model, excluding the forces that govern motion. Some models like map-based models (models that consider map information into the system, Figure 1.5(a)) that do not consider dynamic agents and that are static extensions of the previously defined, are called Static models. Instead, a dynamic model considers external forces and the context of the environment (Figure 1.5(b)). For example, a robot that perceives abstract attractive forces from the desired end location while perceiving repulsive forces from the obstacles can be localized to be in a very changing context;

- *Multi-model approaches*: to describe the complex motion behavior of the dynamic agents, different motion modes are taken into account, including the map and dynamic environment-based models.

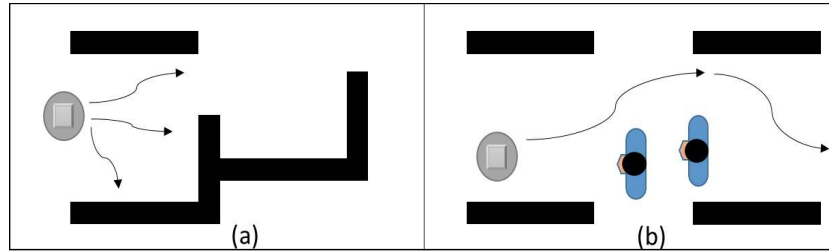


Figure 1.5: Two Physics-based approaches proposed in [80]: (a) Map-based model that does not consider dynamic agents but mainly map information. (b) Dynamic environment-based model that considers both external forces and context of the environment.

Physics-based approaches are very helpful in static environment or where human motion can be modeled via mathematical functions.

Pattern-based approaches

These types of approaches exploit the power of data by deploying Sense-Learn-Predict paradigm. According to the techniques used, they can be divided into the following types:

- *Sequential models*: they are built on the assumption that the current position and velocity (Figure 1.6(a)) can be expressed starting from the data of the previous state's statistical observations (i.e., local transition patterns, topological maps, Voronoi graphs). Most recent techniques used neural networks and Long Short-Term Memory (LSTM);
- *Non-sequential models*: they intend to learn human motion patterns by clustering trajectories observed over a long time. They want to exploit unsupervised clustering techniques to predict human future trajectories in social settings (Figure 1.6(b)).

Pattern-based approaches work well for large environments where dynamics are unknown and multiple humans are involved in the scene.

Planning-based approaches

These types of approaches build on the Sense-Reason-Act paradigm. It makes reasoning about the current state of human motion and acting accordingly. In

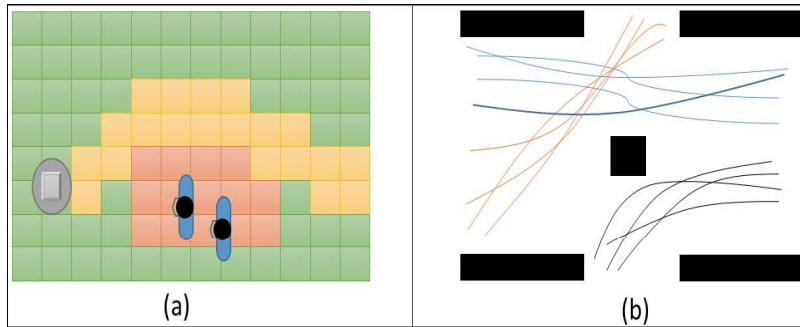


Figure 1.6: Two Pattern-based approaches proposed in [80]: (a) Sequential model. (b) Non-sequential model.

such a way the probability of choosing the best possible navigation path for the robot increases a lot. According to the order of reasoning and acting phases, these approaches can be classified into the following types:

- *Forward planning-based approaches*: they use a predefined function for planning both motion and path (Figure 1.7(a)). Mostly used with probabilistic dynamic models, they consider distances between the robot and the final goal location as a metric to predict the next following state of humans in the context scene;
- *Inverse planning-based approaches*: they compute a cost function by observing humans navigating in the current scene by exploiting various imitation techniques (Figure 1.7(b)). The robot trajectory is selected by transforming this problem into an optimization problem dependent on environment semantics, cost functions, and human interactions.

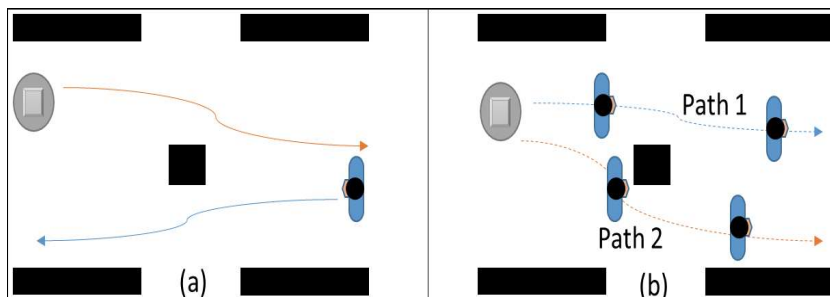


Figure 1.7: Two Planning-based approaches proposed in [80]: (a) Forward planning. (b) Inverse planning.

Planning-based approaches are extremely suitable when robots and human goals can be explicitly defined in an environment.

1.1.4 Context-Aware Mapping

Context-Aware Mapping technology helps the transition from industries built around a human-centered social setting to a human-robot shared social setting. To do so, this technology gives interpretation and helps to understand the robot's geometrical measures, data, and spatial relations to human environments. The context-aware mapping can be divided into Semantic Mapping and Social Mapping.

Semantic Mapping

A robot equipped with Semantic Mapping software uses high-level modalities to formulate its geometrical interpretation. It can accomplish tasks like Optical Character Recognition (OCR), object recognition, and distance measurements appending features understandable by humans on top of metric maps. In such a way, the geometrical interpretation is enhanced and communicated to humans.

Since a robot can work both in indoor and outdoor environments, Semantic Mapping is a technology that can work in both cases:

- *Indoor single scene*: data are acquired by a Red Green Blue-Depth (RGB-D) camera and then they are transformed into a different color space to perform computations like the Conditional Random Field (CRF) model, followed by forest model to execute segmentation of the scene [37];
- *Outdoor scene*: stereo RGB-D cameras perform feature extraction and segmentation at multiple scales. Typically, multimodel approaches are very useful to merge different Field of Views (FoVs) with CRF to have, in general, a very accurate and human-understandable representation of the current scene [40]. As shown in Figure 1.8, each element in the scene (i.e., vehicles, road, vegetation, etc.) is labeled with a label to associate it with the corresponding class.

Social Mapping

As soon as Semantic Mapping has been established, the next step is to take into account human factors. The main ones are sociability, naturalness, safety, and comfort. Social Mapping emphasizes the human-robot coexistence and robot navigation in human-robot shared environments:



Figure 1.8: Semantic segmentation with class labels [80].

- Safety and visibility are the key factors in forming social map. Respectively, the former focuses on the distance between the robot and human, while the latter is built on the criterion that all robots must be inside the field of view of humans, so that no surprises or unexpected behaviors can occur in humans (Figure 1.9) [80];

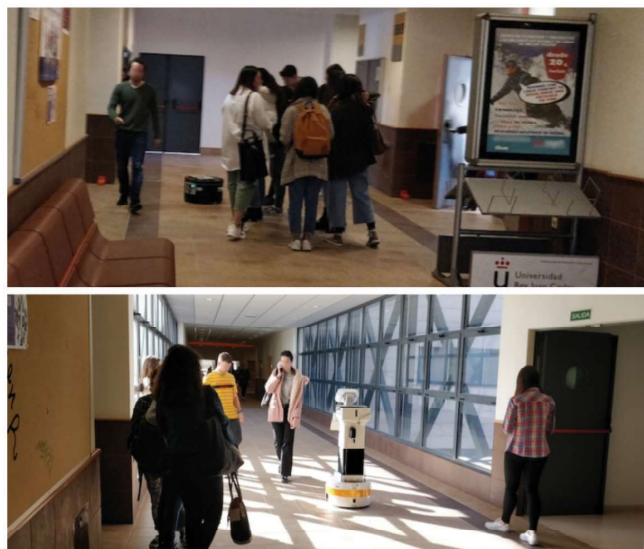


Figure 1.9: Social mapping experiment in an indoor environment proposed in [80].

- SFM is one of the most chosen types of analysis to shape social maps. It aims at explaining and predicting human trajectories with the help of social forces, modeled to represent neighboring obstacles and nearby humans as repulsive forces.

Moreover, customized cost functions that weight human responses can be applied to increase human trust in robots [80].

1.2 Core challenges

Exciting challenges arise day-by-day given the daily spreading of the use of robots. They are present in industrial environments, where automated robots are programmed to perform repeatedly the same set of operations in a closed workspace without nearby humans, and also in human-robot shared contexts. In socially aware navigation, there are the following core challenges according to [59]:

- *Planning challenges*: How should a robot plan its motion through a crowd of navigating humans to ensure safety and efficiency?
- *Behavioural challenges*: What types of social signals can be inferred from human behavior and what types of behaviors should a robot exhibit to ensure social norm compliance?
- *Evaluation challenges*: How can we evaluate a social robot navigation system?

1.2.1 Planning Challenges

Planning Challenges of Social Navigation have been partitioned in literature into two classes: Decoupled Prediction and Planning and Coupled Prediction and Planning. As can be seen, this partition is based on the interaction between the two fields of interest in this subject.

Decoupled Prediction and Planning

These challenges can be further classified based on dominant trends in research:

- *Humans as Dynamic, Non-responsive Obstacles*: humans are treated as non-reactive and dynamic obstacles without a very precise formalization of their social interactions. Some examples are the works proposed in [12], [29], [68];

- *Motion and localization Uncertainty*: the idea is to infer where the agent can be rather than where the agent is. This type of inference reasoning has been boosted by contexts of crowds of humans where detecting people is more difficult due to occlusion and chaotic people motions.

Coupled Prediction and Planning

These challenges are fully covered by the two main trends of cooperation type, capturing multi-agent interactions in crowded domains, where the agents are the robot and the surrounding humans:

- *Explicit Approaches*: they exploit the structure of multi-agent collision avoidance and agent models. For example, they employ explicit predictions of human behavior, typically in the form of trajectories, probability distributions, or expected occupancy maps;
- *Implicit Approaches*: they exploit the principles of cooperative collision avoidance without imposing explicit constraints on the structures. They work at a higher level w.r.t. explicit ones.

1.2.2 Behavioral Challenges

The Behavioral Challenges concern the core aspects of a successful navigation with a focus on safety and efficiency. To reach the level of abstraction required for developing realistic behavioral software, the more common and general difficulties of behavioral implications of interactions between agents must be carefully studied.

Behavioural Interactions

These interactions can be categorized with increasing order of social signal “richness”:

- *Proxemics*: as already explained in Section 1.1.2, in other words it describes the space around a pedestrian⁵;

⁵A human walking in an environment.

- *Intentions*: they represent rich cues that offer hints about the future behavior of an agent. For example, for a robot, an intention may be communicated to humans through some particular gesture of the arm, body posture, gaze, etc. These behaviors are fundamental to communicate the behavior, conveying the actual and future state of the agent;
- *Formations*: the most popular research on pedestrian spatial behavior is on grouping, which is a synchronization process where a few individuals form a group. That group can form a static or dynamic setting: in the former humans do not move and form interactions, while in the latter humans are moving. Both static and dynamic group formations have to be respected by mobile robots.

1.2.3 Evaluation challenges

Another core challenge regards the evaluation of the performances obtainable by methodologies.

Metrics

The Evaluation Challenges can be faced by measuring some aspects like the following:

- *Navigation Success Rate*: how often an agent reaches its goal destination;
- *Path Efficiency and Optimality*: the quality of the trajectories;
- *Safety/Collision Avoidance*: the number of constraint violations. For instance, the number of collisions;
- *Behavioral Naturalness*: even if it is not directly connected to planning performances, it represents a fitness score for a robot navigating in a human-populated environment. One of the most general and common approaches is to take a dataset of recorded human pedestrians. After, the score will represent the distance between an agent's plans and the trajectories saved in the data. According to the algorithm the score is computed, Average Displacement Error (ADE) or Final Displacement Error (FDE) can be measured. The former is the average of Euclidean differences between temporally aligned points in the paths given by a navigation algorithm and the

ground truth path of the dataset. Instead, the latter is the displacement error only at the final time step.

All the above metrics can evaluate the performances of a single agent. Then, a multi-agent study can be performed comparing each single-agent study.

A more schematic list of the common metrics proposed and detailed in [59] can be found in Table 1.1.

Name	Description
Arrival rate	How often an agent successfully reaches the goal
Path length	The distance traveled by an agent
Collision rate	Frequency of collision per agent
Failure rate	How often an agent fails to reach the destination
Average time to goal	Average time spent by an agent to reach the goals
Social scores	Indices to measure human comfort such as safety for the path planned by the robot or the number of social constraints violated
Average acceleration	Acceleration averaged per time
Average energy	Integral of the squared velocity of an agent averaged per time
Path irregularity	Total amount of displacements beyond planned path
Path efficiency	Ratio between straight line path and planned path
Time spent per unit path length	Average time spent to move for one unit of path length
Topological complexity	Amount of entanglement among agents' trajectories
Speed efficiency	Ratio between nominal and actual speed

Table 1.1: Evaluation metrics for social navigation proposed in [59].

1.2.4 Open Problems

For the challenges above several fundamental problems are still open:

- *Modeling Interactions in Crowded Environment:* as soon as the number of humans in a context increases, problems and difficulties increase a lot. For example, if a robot navigates inside a dense crowd, it is very difficult for it to socially navigate in a smooth way exploiting the most common techniques such as SFM, without accounting for human cooperation;
- *Behavioral Design:* since the infinite number of possible contexts, it is both hard to code to embed hand-crafted behaviors in the robot software and to train navigation models that can perform well in a wide variety of contexts and environments;
- *Role of Context:* taking into account some main properties of the context (e.g.: the shape of the space, the timing, the scene, and the grouping behavior) can lead to both pros and cons. Indeed, some unexpected behaviors or

problems can be faced by both embedding hand-crafted changes and learning algorithms in the robot software. However, this can lead to a loss of generality;

- *Better Behavior Understanding*: this is the biggest behavioral challenge in social navigation [59]. At the current stage, research in pedestrian motion understanding takes a trial-and-error-based approach. Indeed, because of the limited knowledge available, researchers tend to use Deep Learning techniques to implicitly model pedestrian behaviors. In such a way, they can return in output behavioral outcomes (i.e., trajectory predictions, navigation actions, etc.). Yet this still does not produce precise knowledge in this field of research;
- *Behavior Importance Evaluation*: humans behavior analysis is a various behavioral set of aspects that potentially benefit social navigation. In addition, in simulation environments, pedestrians' behaviors are not defined in a precise way, trying to simulate as randomly as possible the widest variety of situations. Until now, this can be the only way to analyze the practical effectiveness of solutions in real-world environments;
- *Limitations of Existing Simulation Practices*: since simulating crowd navigation is a challenging task as it requires assumptions, abstraction does not consider too many interactions that naturally may arise in real life;
- *Towards a Benchmarking Protocol*: in the State-of-the-Art there is a large lack of standardized experimental benchmarks. The research community needs to converge towards a standard definition to specify all necessary experimental features: for instance exact definition of the task the robot has to perform, types of experimental platforms with pros and cons, the number and the role of humans, the context and the environment, the metrics to use. In such a way the comparison among different approaches should be facilitated and more consistent;
- *Formal Verification*: it is worth noting that even if a real-world benchmark would have been established, it could be not enough representative about the performances. Indeed, any benchmark would still inevitably at best capture a small sample of representative real-world context and interactions. Therefore, adding methods for formal verification to benchmarks would

strengthen performances. Unfortunately, in realistic settings more deep research is still necessary.

1.3 Aim of the thesis

State-of-the-art approaches have made progress in human-robot shared tasks, but challenges arise when robots operate in human crowded environments. The key issue is to align robot behavior with human perception, both in how people perceive the robot and how the robot perceives and interacts with people. Active Perception tries to face both sides of this problem through the involvement of behavior selection (e.g., get close to the possible target, move around to add information about the environment, get close to a zone where there is small certainty about what can be found there, ...) to enhance information gathering in a specific environment and context. Robots move and explore their surroundings by using their sensors to sample data and construct an understanding of the environment. In active perception, the interpretation of sensors' data is closely linked to the behaviors needed to capture that data, with action and perception tightly coupled. In the people-aware navigation, challenges include effective robot motion for people detection and tracking or re-identification, dealing with limited visibility in crowded areas, integrating data from various sensors, and addressing the computational load of different components and modules in robots (e.g., perception, detection, choice of an action in line with the dynamics of the social context, ...).

The thesis aims at developing different possible solutions based on model-based vs. RL approaches for the Active Perception task, with a focus on crowded human environments, exploiting only data incoming from 2D LiDAR. More precisely, the robot has to improve the confidence in people detection through the execution of defined actions, while trying to find new people in different scenarios. Moreover, the robot should observe social behaviors during navigation since real robots are the final users of these works for real applications, where interactions with real humans are involved.

1.4 Structure of the thesis

This master thesis is organized into seven chapters, each contributing to a comprehensive understanding of the research on the active perception topic, with

particular attention to social navigation in crowded environments.

In this introductory chapter, the research problem, its significance, and the motivation behind the study are presented. The main research questions and objectives are outlined, setting the context for the whole thesis.

Chapter 2 delves into the existing literature and provides an extensive review of relevant studies, methodologies, and technologies, with a comparison of their most significant aspects. The focus is mainly on geometrical, SFM, and learning approaches. After a take-home message to highlight the most remarkable aspects, the last section discusses the gap between active perception techniques and what is the goal of this thesis.

Chapter 3 introduces the tools and methods employed in this research. It details the hardware and software utilized, including the 2D LiDAR sensor and the most important packages. Additionally, it is discussed the experimental setup and data collection procedures.

Chapter 4 focuses on the preprocessing steps necessary for an effective continuation of the work. It covers techniques for people detection using 2D LiDAR data and the classification of laser points to distinguish between different objects in the environment.

Chapter 5 is the core of the thesis, where the concept of socially active perception is explained and elaborated upon. This chapter explores how the system perceives and responds to social cues and human interactions to behave actively and to successfully perceive environments. Different types of approaches are deeply analyzed, from a custom-defined policy to RL methods.

Chapter 6 presents the experimental design and the results obtained from various tests. It includes quantitative and qualitative analyses to evaluate the performances of the proposed active perception systems in human crowded environments.

The final chapter (7) summarizes the key findings of the research and addresses the research questions and objectives outlined in Chapter 1. It collects the implications of the results and their contributions to the field of active perception and social navigation. Additionally, recommendations for future research and potential applications are discussed.

Chapter 2

State-of-art

This chapter illustrates the state-of-the-art approaches in the social-aware navigation with particular attention to the active-perception. The proposed overview aims at highlightening the pros and the cons of the different approaches (geometrical, social force model-based, and learning-based) from the simplest to the most recent and advanced methods. Finally, the chapter ends with the take-home messages from the review of the literature.

2.1 Geometrical Approaches

All the approaches described in this section rely on the concepts of personal space, Gaussian functions, and modified costmaps, in conjunction with geometrical formulations. In this section three main works in the application of geometrical methods in SARN are presented.

2.1.1 RGB-D and Laser Data Fusion based Human Detection and Tracking

In [86], authors propose a framework to let mobile robots distinguish humans from other types of regular obstacles. Moreover, they estimate human states (e.g., human positions, and motion). This is achieved by fusing multiple sensor data to extract the information needed. Then, extended personal spaces are modeled and incorporated into the kinodynamic Rapidly-exploring Random Trees (RRT) motion planning system to get the acceptable¹ trajectories for the robots.

¹Here the term “acceptable” refers to the final trajectories that are comprehensible and consistent in terms of human safety and comfort in social environments.

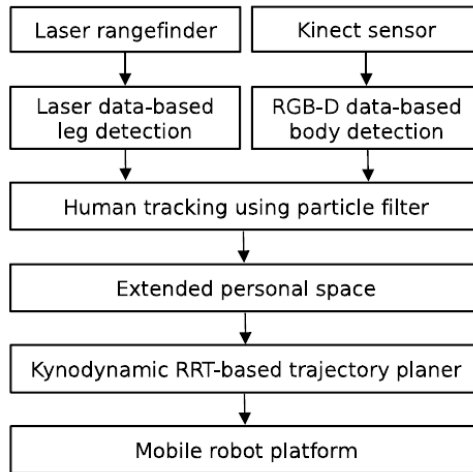


Figure 2.1: A flowchart of the proposed socially aware robot navigation system proposed in [86].

The proposed framework in Figure 2.1 can be divided into three stages:

1. *Human detection and tracking*: multisensor fusion technique to extract and track human states.

The robot used in [86] is equipped with a Microsoft Kinect sensor (Red Green Blue (RGB) camera and depth sensor) and a laser range finder. As depicted in Figure 2.2, data collected are then processed to produce for both sensors a parameter set (a set of $h_l = (x_l, y_l)$ for the laser and a set of $h_b = (x_b, y_b)$ for the Kinect sensor) that indicate the detected humans. To complete the tracking, the fusion of data is done by performing a particle filter [8] with Global Nearest Neighbour data association [50]. The output of the human tracking system is a collection of human states² $p_i = (x_i^p, y_i^p, \theta_i^p, v_i^p)$ which is used to extend the personal space in the next stage;

2. *Extended Personal Space (EPS)*: human comfort and safety guaranteed by EPS.

To model the space around a human it is used a two-dimensional Gaussian function with the maximum at the center that gradually descends away from it. This concept is based on the Hall personal space (Figure 2.3 (a)), extended as the concept of basic personal space (Figure 2.3 (b)) divided into the frontal and back area around a human p_i .

²Each human state is represented by the position point (x_i^p, y_i^p) in which the human is located, its orientation θ_i^p , and its speed v_i^p .

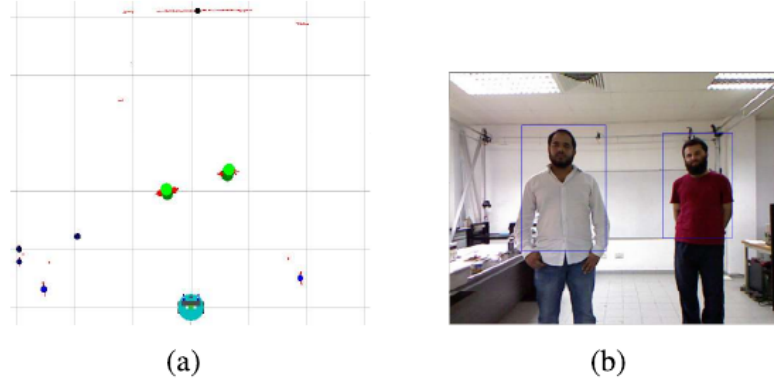


Figure 2.2: The result of the human detection method proposed in [86]: (a) laser-based human detection, (b) RGB-D based on human body detection.

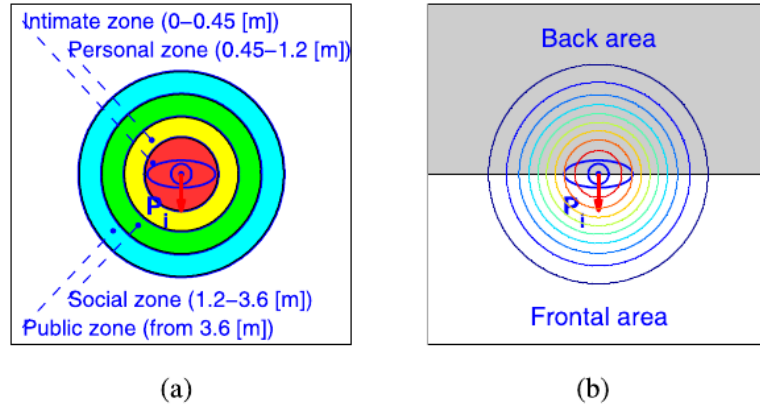


Figure 2.3: (a) Hall model, (b) Basic personal space using two-dimensional Gaussian function proposed in [86].

Since robot states are defined as $r = (x_r, y_r, \theta_r, v_r)$, the relative pose and motion between humans and robots can be formalized as in Figure 2.4.

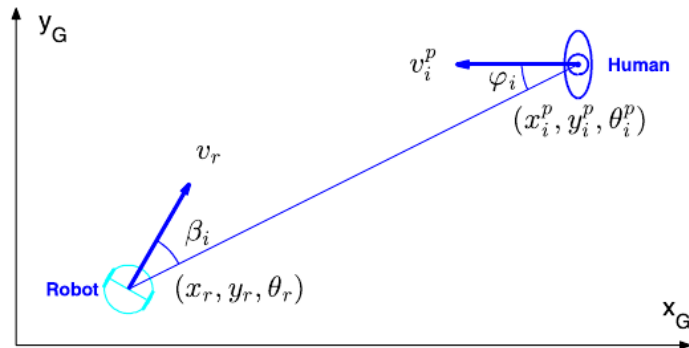


Figure 2.4: Related pose between a person and a robot [86].

At this point, the N EPSs of all people detected by the robot are represented by the function $F_{eps}(x, y) = \max(f_1^{eps}(x, y), \dots, f_N^{eps}(x, y))$ where f_i^{eps} is the

model of the EPS around person i ;

3. *System integration*: final legible trajectories in output by kinodynamic RRT-based motion planner.

This is done as represented in the data flow diagram in Figure 2.5.

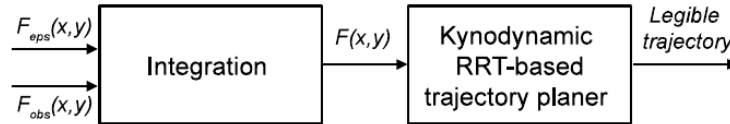


Figure 2.5: Data flow diagram of the system integration proposed in [86].

The EPS function $F_{eps}(x, y)$ is integrated with the obstacle function $F_{obs}(x, y)$. This last function takes into account the static and dynamic obstacles detected by the laser rangefinder in the environment. The final local cost function in the proximity of the robot is $F(x, y) = \max(w_1 f_{eps}(x, y), w_2 f_{obs}(x, y))$ where the couple (x, y) is the coordinate of points around humans, w_1 and w_2 are the weights of the two types of functions. It is this $F(x, y)$ function that is used in input to the kinodynamic RRT motion planning [52]. This procedure is formalized in Algorithm 1 [86].

Algorithm 1 Kinodynamic RRT

Input: $q_{start}, q_{goal}, K, F(x, y), \Delta t, MP$

Output: Robot Trajectory: $traj$

- 1: Initialize τ with q_{start} ;
 - 2: **while** $N_{nodes} < K$ **do**
 - 3: $q_{rand} \leftarrow \text{randomState}()$;
 - 4: **if** $\neg \text{checkCollision}(q_{rand}, F(x, y))$ **then**
 - 5: $q_{near} \leftarrow \text{nearestNeighbor}(\tau, q_{rand})$;
 - 6: $q_{new}, u_{best} \leftarrow \text{extend}(q_{near}, q_{rand}, \Delta t, MP)$;
 - 7: **if** $\neg \text{checkCollision}(q_{new}, F(x, y))$ **then**
 - 8: $\tau.\text{addVertex}(q_{new})$;
 - 9: $\tau.\text{addEdge}(q_{near}, q_{new}, u_{best})$;
 - 10: **if** $q_{new} \in q_{goal}$ **then**
 - 11: $traj \leftarrow \text{extractTrajectory}(\tau, q_{new})$; **return** $traj$;
 - 12: **end if**
 - 13: **end if**
 - 14: **end if**
 - 15: **end while**
-

Simulated Experiments and Results

Authors of [86] implemented their proposed framework using Matrix Laboratory (MATLAB)³ and C/C++ programming language with Robot Operating System (ROS), Open Source Computer Vision Library (OpenCV)⁴ and Point Cloud Library (PCL)⁵. Tests ran on an Intel core i7 2.2 GigaHertz (GHz) laptop. All tests use the same initial start and goal poses, expecting the robot to move avoiding obstacles and humans in the surrounding area, testing also the reliability of the final path. Tests consider more situations: a human standing in front of the final path. Tests consider more situations: a human standing in front of the mobile robot, a robot navigating while a person is moving and a group of two people are standing in front of the robot (Figure 2.6).

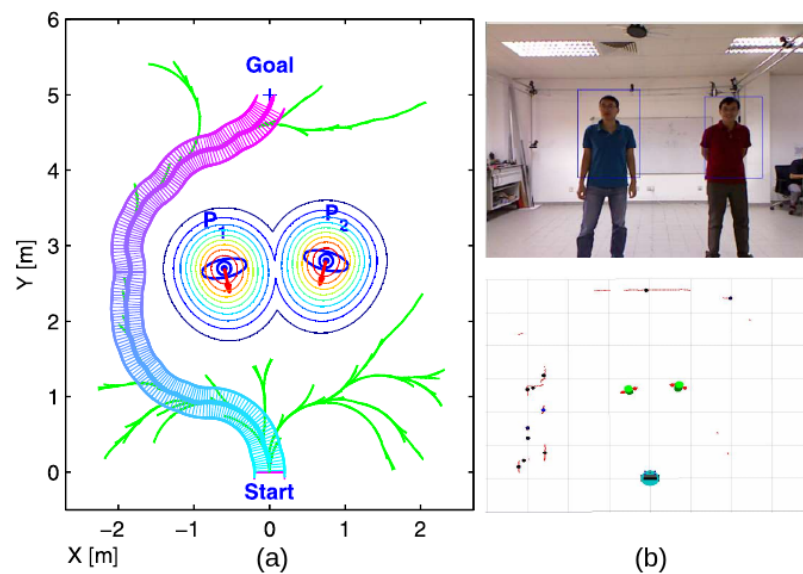


Figure 2.6: A group of two people: (a) the extended personal space shown in contour, the RRT tree (in green color) and the trajectory of the robot generated by the kinodynamic RRT motion planner; (b) the human detection using RGB-D data and the human leg detection using laser data proposed in [86].

This framework achieves good results in terms of human position and motion states, controlling the robot to navigate in a good way. However, this method is suitable and tested in very sparsely crowded contexts. Indeed, in the case of crowds, extended social spaces around humans can force the robot not to move since it is surrounded without any free way.

³For further details: <https://it.mathworks.com/products/matlab.html> .

⁴For further details: <https://opencv.org/> .

⁵For further details: <https://pointclouds.org/> .

2.1.2 Dynamic Social Zone (DSZ) based Mobile Robot Navigation

In [84] an effective Dynamic Social Zone (DSZ) based navigation method for mobile robots is proposed. Here the emphasis is on human safety in social environments. With respect to the work presented in Section 2.1.1 ([86]), the work in [84] allows to plan trajectories to avoid undesired physical contact with humans, considering also psychological aspects and social constraints. Moreover, another important difference with [86] is that this safety framework is applied to mobile robots to navigate while avoiding humans to reach an appropriate approaching position.

The system architecture of the extended navigation scheme for mobile robots in social environments consists of a conventional navigation scheme (lower part of Figure 2.7) with the addition of the human comfortable safety framework (in cyan color in Figure 2.7). To the four blocks of the traditional navigation, the human framework adds the ability to extract human characteristics for the development of the DSZ. The output of DSZ is a human-aware decision that guarantees human safety and comfort in both avoiding and approaching people.

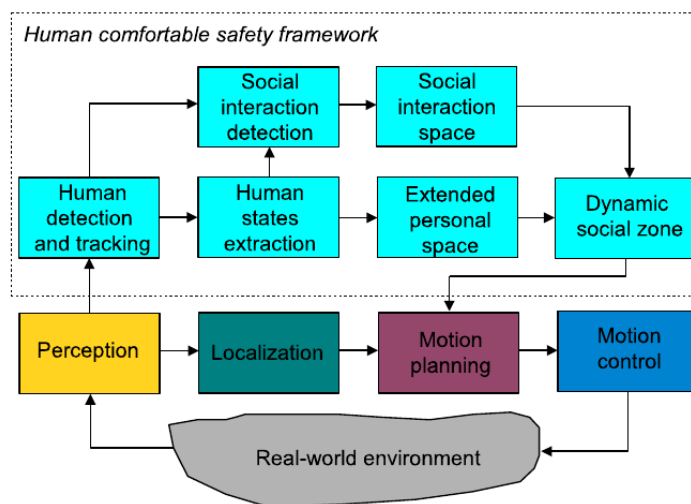


Figure 2.7: The extended navigation scheme for mobile service robots proposed in [84].

The human comfortable safety framework of [84] is composed of six blocks:

1. *Human states*: they provide spatial and temporal characteristics of humans such as human positions, orientations, velocities, hand poses (for human actions), fields of view, and human gaze directions (for the direction of social attention of a human). All of them help to identify group interactions;

2. *EPS*: as in Figure 2.3, this concept is used and it is enriched with information like human hand poses and affected by other factors such as social situation, gender, age, personality, etc;
3. *Social Interaction Detection*: social interaction detection can be classified into two types. The first, about human-object interaction detection, is the key to define the interaction space between humans and interesting objects. The second, instead, has an essential role in detecting human groups. More precisely, the respective algorithm in [84] outputs the number of human groups and their center points, where each human group is called g_i with $0 \leq i \leq (k - 1)$ in case of k groups;
4. *Social Interaction Space (SIS)*: this type of space is built by incorporating the space of social interactions detected with the use of Gaussian distributions and parameters that were properly tuned;
5. *DSZ*: it is defined as dynamic SIS around a human, a human group, or human-object interaction. It derives by incorporating the social interaction spaces and the EPSs as in Equation 2.1:

$$F_{dsz}(x, y) = \max(w_3 F_{eps}(x, y), w_4 F_{sis}(x, y)) \quad (2.1)$$

In this way, robots are not allowed to navigate too much next to humans, to avoid psychological discomfort and increase physical safety. DSZ is then passed in input to Algorithm 2 [84];

6. *Approaching humans*: to accomplish this task, the estimation of the approaching area and the computation of the final goal pose must be done. There are many context-dependent factors to consider. However, the conditions to satisfy in all cases are that a mobile robot must end outside the DSZ and in the human FoVs. Figure 2.8 shows the procedure to compute the final approaching goal pose (Algorithm 2 [84]). The final robot's pose for approaching people is computed as based on the current robot position and it is considered more likely the one in the center of the approaching areas (Figure 2.8).

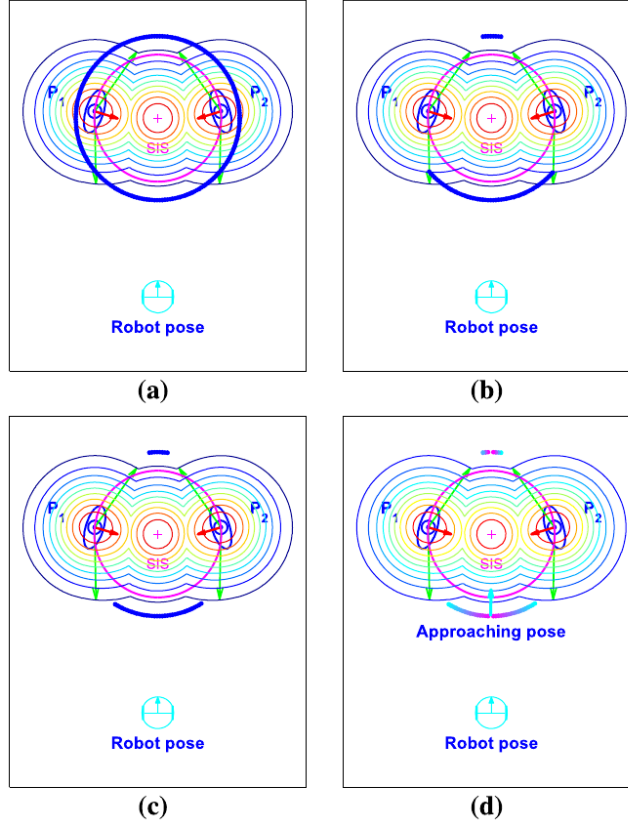


Figure 2.8: Estimation of the approaching pose for the robot: (a) the approaching area is the thick blue circle; (b) the approaching areas filtered by the human field of view; (c) the approaching areas filtered by the dynamic social zone; and (d) the robot's approaching pose as the central point of the approaching area. The figure is taken from [84].

Algorithm 2 Estimate Approaching Goal Pose

Input: $g_k = (x_k^g, y_k^g)$, $p_i = (x_i^p, y_i^p)$, A_{out} , (x_r, y_r)

Output: Approaching goal pose $q_{goal} = (x_q, y_q, \theta_q)$

- 1: **for** each approaching area A_{outj} in A_{out} **do**
 - 2: $L_j \leftarrow \text{length}(A_{outj})$
 - 3: $(x_{cj}, y_{cj}) \leftarrow A_{outj} \left(\frac{L_j}{2}, \frac{L_j}{2} \right)$
 - 4: $d_j \leftarrow \sqrt{(x_r - x_{cj})^2 + (y_r - y_{cj})^2}$
 - 5: **end for**
 - 6: Find the smallest d_j
 - 7: **if** d_j is the smallest **then**
 - 8: $(x_q, y_q) \leftarrow (x_{cj}, y_{cj})$
 - 9: **if** a human p_i **then**
 - 10: $\theta_q \leftarrow \text{atan2}(y_i^p - y_q, x_i^p - x_q)$
 - 11: **else if** a group of humans g_k **then**
 - 12: $\theta_q \leftarrow \text{atan2}(y_k^g - y_q, x_k^g - x_q)$
 - 13: **end if**
 - 14: **end if**
 - 15: **return** $q_{goal} \leftarrow (x_q, y_q, \theta_q)$
-

Similarly to the methods proposed in [86], in [84] a synthesis function is used (Equation 2.2) and its block diagram and data flow of incorporated DSZ into motion planning system is in Figure 2.9.

$$F(x, y) = \max(w_5 F_{dsz}(x, y), w_6 F_{obs}(x, y)) \quad (2.2)$$

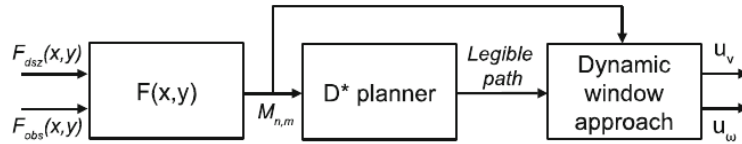


Figure 2.9: Block diagram of incorporating the DSZ into the motion planning system [84].

For experiments, authors considered the robot as a human to trace psychological safety in terms of relative poses $(x_i^p, y_i^p, x_r, y_r, \alpha_i, \beta_i)$ and relative motion (v_p, v_r) humans-robot.

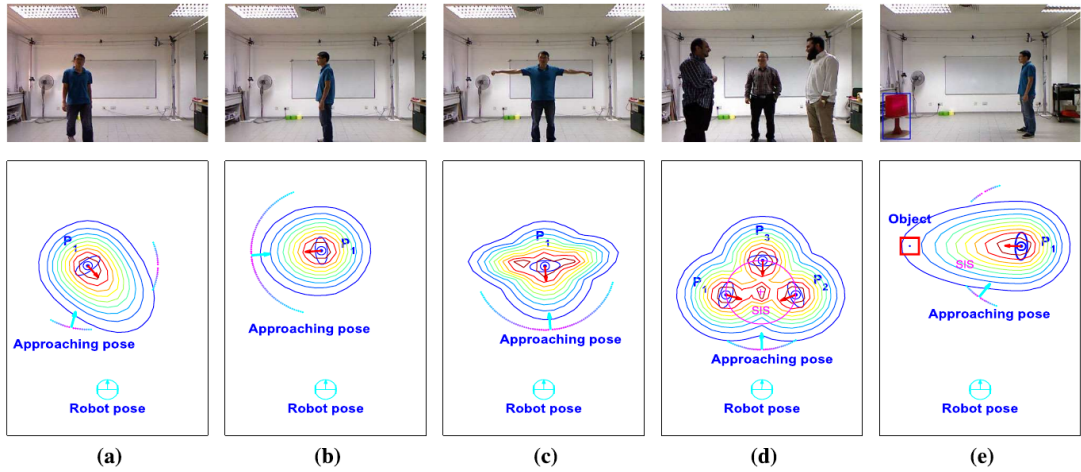


Figure 2.10: The result of the human safety framework in different scenarios: the first row shows the images from robot's camera, and the second row shows the results of the dynamic social zone and approaching pose of the robot corresponding to the scenarios in the first row. Each P_i indicates a person. From left to right: (a) a moving person, (b) a head orientation, (c) a person with a hand pose, (d) a group of three people, and (e) a human-object interaction. The figure is taken from [84].

Implementation and tests done in [84] used a MATLAB-based simulation. Then, the framework has been tested in a real environment, in more contexts.

Simulations consider the following cases:

- Humans are treated like obstacles: the robot never shows social behaviors since it is not endowed with the capacity of differentiating between humans and obstacles;

- Use of the EPS: the robot takes into account humans using only their EPSs. Sometimes the robot navigates in a good way staying far enough from humans, other times it passes in between humans and objects they were interacting with;
- Use of the DSZ: the robot takes into account humans, hence using DSZ. The robot always navigates showing social behaviors staying far enough from humans, without passing in between humans and objects they were interacting with.

Then, to have more reliable results, the authors tested the system using the real robot. Their focus was on a series of experiments to demonstrate that DSZ is very effective in SARN in terms of human comfort safety:

- DSZ: as illustrated in Figure 2.10, DSZ can adapt very well to variations of contexts, human states, and social interactions, providing an interesting and correct approaching pose in terms of social behaviors;
- Avoiding a human group and approaching a human: the comparison between the same case with and without the use of DSZ lets to two very different cases and results, verifying another time the social correctness of the framework;
- Avoiding a human and approaching a human group: as in the precedent point, the difference between the two cases is important, and approaching pose to a group has been proved.

In conclusion, DSZ is dynamically adaptable to variations of social interaction information and single human states. In this way, robots are capable of avoiding and approaching both single and groups of humans. Nevertheless, real-time, powerful, and robust human detection and tracking modules are needed to improve performances in a significant way. Also tests performed in this work have involved only static or semi-static environments, populated by a few people. In crowded environments, the robot can end in a local minima without the possibility to find a way out. This aspect is a limitation of this method and the one presented in Section 2.1.1.

2.1.3 Approaching Pose Prediction

To overcome the weaknesses of the work in Section 2.1.2, the authors of [85] propose a unified human approaching framework as an extension of the work in [84]. In this way, [85] aims at enabling a mobile robot to approach both stationary and dynamic humans and groups of interacting humans in a socially acceptable manner. The main change that allows to handle dynamic humans with respect to the previous version is the separation of the approaching pose estimation block out of the one for DSZ to build a new functional block for approaching pose prediction. Moreover, the author of [85] merged human status information into the EPS and the velocity of moving social groups into the SIS.

The newly updated system architecture of the SARN system for mobile robots to approach humans in a predicted way is depicted in Figure 2.11. It is composed of the following blocks:

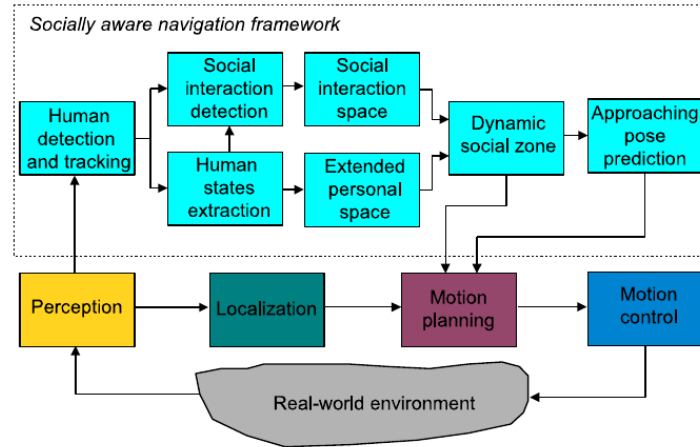


Figure 2.11: The Extended navigation scheme for mobile service robots proposed in [85] is composed of two main parts: 1) a conventional navigation scheme and 2) a socially aware navigation framework (in cyan color).

1. *Human Detection and Tracking*: it is based on the system in [86] (please refer to Section 2.1.1);
2. *EPS*: it is computed similarly to [84];
3. *SIS*: similarly to [84], it is calculated with more emphasis on Social Group Detection and Social Group Modeling;
4. *DSZ*: like in [84], it was formulated according to the Equation 2.3:

$$F_{dsz}(x, y) = \max(w_1 F_{eps}(x, y), w_2 F_{sis}(x, y)) \quad (2.3)$$

where $F_{eps}(x, y) = \max(f_1^{eps}(x, y), \dots, f_N^{eps}(x, y))$ and

$F_{sis}(x, y) = \max(f_1^g(x, y), \dots, f_K^g(x, y))$, with N the number of humans next to the robot and K the number of human interaction groups;

5. *Approaching Pose Estimation and Prediction*: regarding to the pose estimation there is not something more to add to the requirements and constraints remarked in the correspondent part of Section 2.1.2. Instead, two situations can happen. For static humans, the estimated approaching pose can be used $q_{goal} = (x_q, y_q, \theta_q)$, while for moving humans two steps are required: the future human pose must be predicted and then the new approaching pose of the robot must be estimated (Figure 2.12). The method proposed in [85] suggests the use of a Kalman filter [46] with a constant velocity-based motion model of the humans to get a predicted pose that is passed in input to the proposed framework to accomplish the task. The updated approaching pose of the robot after an interval Δt is $q_{goal}(t + \Delta t) = (x'_q, y'_q, \theta'_q)$.

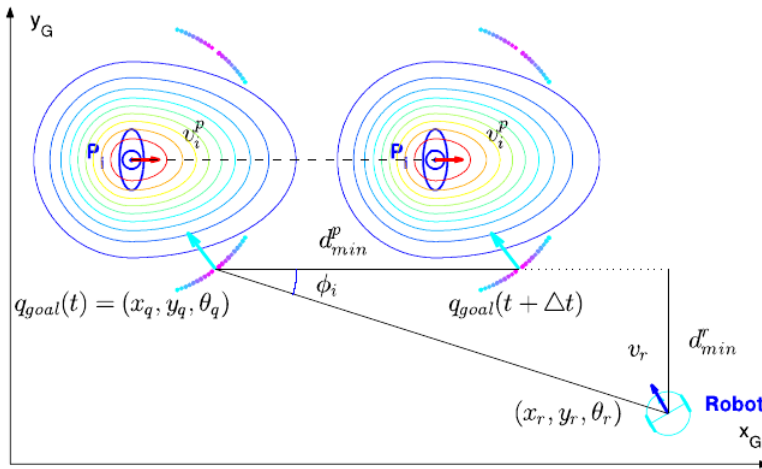


Figure 2.12: Approaching pose prediction of a moving person p_i . The current approaching pose is $q_{goal}(t)$ and the predicted approaching pose $q_{goal}(t + \Delta t)$. The figure is taken from [85].

Analogously to [84], the synthesis function (Equation 2.4) is used, and its block diagram and data flow is in Figure 2.13.

$$F(x, y) = \max(w_3 F_{dsz}(x, y), w_4 F_{obs}(x, y)) \quad (2.4)$$

Implementation has been done using MATLAB, C++ programming language, OpenCV, PCL, and ROS. The unified framework proposed in [85] has been tested both in simulated and real environments.

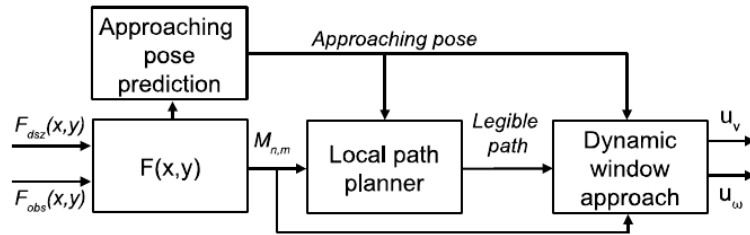


Figure 2.13: Pipeline of the framework proposed in [85] that incorporates the DSZ into the motion planning system.

About simulations, the challenges authors faced are the following:

- Humans are treated like regular obstacles: even though the robot does not physically hurt humans, humans do not feel comfortable and safe;
- Use of the DSZ: removing the approaching pose prediction block, the robot does not move too close to humans and SISs during navigation. However, it is not able to estimate the approach pose both for single humans and groups of humans socially and properly;
- Use of the approaching human framework: the robot can behave in a socially compliant way, ending successfully in all ten social contexts of the tests proposed by authors.

The two experiments performed by authors regard real-environment tests:

- Approaching stationary humans divided into four particular test cases, performances are good with a possibility to improve them, especially using more precise sensors. The four cases are about a sitting person, a standing human, a group of three standing people, and an interaction between two humans and an interesting object;
- Approaching dynamic humans: this test is the most interesting and it includes three scenarios represented in Figure 2.14. The first row in the figure shows a third-per then the new approaching pose must be estimated son view of the scenarios. The second row shows snapshots of the ROS Visualization (RViz) visualization at the instance when the robot observed the predicted approaching goal pose and the humans moved toward their predicted positions. Moreover, this row shows the human poses, the predicted approaching pose, the approaching areas, the DSZ, and the planned path of the robot. The third row visualizes the real trajectories of the robot and the

humans in which the current approaching pose $q_{goal}(t)$ and the predicted approaching pose $q_{goal}(t + \Delta t)$, and the DSZ, are highlighted. The robot is very good at predicting future poses of humans and, as a consequence, it estimates the correct approaching poses in all cases.

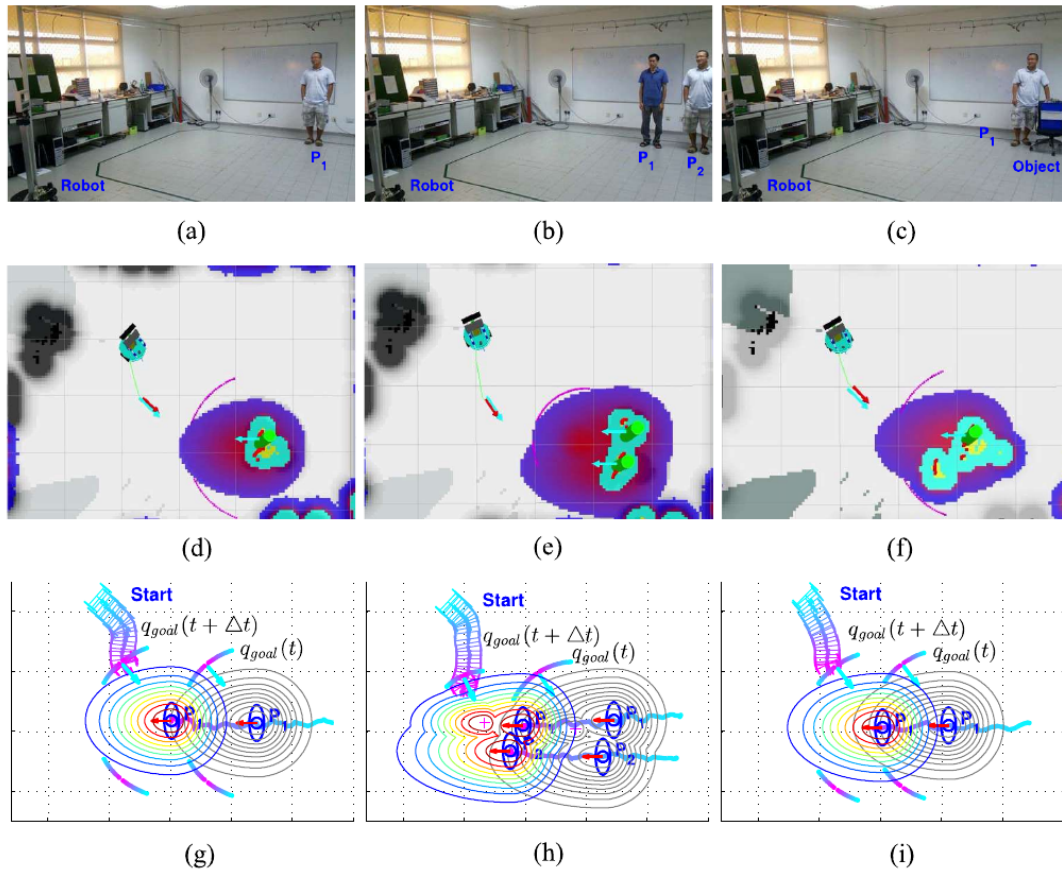


Figure 2.14: Experimental results of the second experiment where the robot approaches dynamic humans as proposed in [85]. (a), (d), (g) Walking person. (b), (e), (h) Group of two walking people. (c), (f), (i) Person moving an object.

This framework is highly capable of enabling the robot to approach humans, independently of their status (standing or moving). Unfortunately, this framework deals with social contexts in shared human-robot environments, but where only a few people are present.

Such approaches described in Section 2.1 seem not to be suited for crowded environments in the SARN research topic.

2.2 Social Force Model (SFM)-based Approaches

In this section, the formulation of the general topic of SFM will be provided before the presentation of one application in socially aware robot navigation research topic.

2.2.1 Social Force Model for pedestrian dynamics

The motion of pedestrians can seem to be chaotic and not very predictable. However, behavioral changes are guided by sensory stimuli that cause a reaction that depends on personal aims. Hence, the motion of walking humans can be described as subject to “social forces”. These are not real forces directly exerted by the environment on the pedestrian body, but they are the concrete cause of why people perform a certain action because they influence position and speed of an agent. Indeed, as a reaction to the perceived environment, a human accelerates or decelerates. This is the reason why one can say that a pedestrian acts as subject to external forces. In other words, social forces $\vec{F}_\alpha(t)$ represent the effect of the whole environment on the behavior of the walking human.

These forces can model the pedestrian behaviors using all measurable quantities like acceleration towards the desired velocity of motion, a certain distance to keep concerning other humans, walls and obstacles, and attractive forces concerning goals.

In this subsection, words like humans, walking humans, and agents refer all to the same concept of an agent that can move taking into account the environment. Therefore, this agent can be a pedestrian, but also a robot navigating in a human-robot shared environment.

As explained in [38], some main features of social forces guide the motion behavior of a human α .

To reach a position goal \vec{r}_α^0 in a comfortable and in the shortest possible way, the points in between start and goal positions generate a polygon shape with edges $\vec{r}_\alpha^1, \dots, \vec{r}_\alpha^n := \vec{r}_\alpha^0$. From the actual position $\vec{r}_\alpha(t)$ of pedestrian α at time t , \vec{r}_α^k is the next edge of the polygon to reach. The desired direction $\vec{e}_\alpha(t)$ of motion is:

$$\vec{e}_\alpha(t) := \frac{\vec{r}_\alpha^k - \vec{r}_\alpha(t)}{\|\vec{r}_\alpha^k - \vec{r}_\alpha(t)\|} \quad (2.5)$$

As a consequence, at every time t the human steers for the nearest point⁶ $\vec{r}_\alpha^*(t)$.

The desired speed v_α^0 , the deviation of the actual velocity $\vec{v}_\alpha(t)$ from the desired velocity $\vec{v}_\alpha^0(t) := v_\alpha^0 \vec{e}_\alpha(t)$ due to collision avoidance or other motivations, lead to reach $\vec{v}_\alpha^0(t)$ with some relaxation/delay time τ_α . All these terms influence the specific force for the acceleration term:

$$\vec{F}_\alpha^0(\vec{v}_\alpha, v_\alpha^0 \vec{e}_\alpha) := \frac{1}{\tau_\alpha} (v_\alpha^0 \vec{e}_\alpha - \vec{v}_\alpha) \quad (2.6)$$

Since the motion of an agent α is affected by other agents, a certain distance must be kept. This depends on the crowd density and the desired speed v_α^0 . This is one of the motivations for the repulsive effects to consider. These repulsive effects from other pedestrians β are formulated as in Equation 2.7:

$$\vec{f}_{\alpha\beta}(\vec{r}_{\alpha\beta}) := -\nabla_{\vec{r}_{\alpha\beta}} V_{\alpha\beta}[b(\vec{r}_{\alpha\beta})] \quad (2.7)$$

In detail, the repulsive potential $V_{\alpha\beta}(b)$ is a monotonic decreasing function of b with equipotential lines having the form of an ellipse directed into the motion direction, since a pedestrian needs space for the next step, which is considered by other pedestrians.

Moreover, b denotes the semi-minor axis of the ellipse such that:

$$2b := \sqrt{(\|\vec{r}_{\alpha\beta}\| + \|\vec{r}_{\alpha\beta} - v_\beta \Delta t \vec{e}_\beta\|)^2 - (v_\beta \Delta t)^2} \quad (2.8)$$

where $\vec{r}_{\alpha\beta} := \vec{r}_\alpha - \vec{r}_\beta$ and $s_\beta := v_\beta \Delta t$ is the step width of pedestrian β .

Collisions with borders of buildings, streets, walls, and obstacles must be avoided. This is the reason why if a border B provokes a repulsive effect, this effect must be defined in the following way:

$$\vec{F}_{\alpha B}(\vec{r}_{\alpha B}) := -\nabla_{\vec{r}_{\alpha B}} U_{\alpha B}(\|\vec{r}_{\alpha B}\|) \quad (2.9)$$

with a repulsive and monotonic decreasing potential $U_{\alpha B}(\|\vec{r}_{\alpha B}\|)$, $\vec{r}_{\alpha B} := \vec{r}_\alpha - \vec{r}_B^\alpha$ and \vec{r}_B^α represents the position of the part of border B that is near to pedestrian α .

In contrast to precedent equations, all regarding repulsive situations, also attractive situations towards target people or objects must be considered. Monotonic increasing potentials $W_{\alpha i}(\|\vec{r}_{\alpha i}\|, t)$ model the attractive effects $\vec{f}_{\alpha i}$ at places

⁶More precisely, the term ‘‘points’’ takes into account areas of space. This because an agent must achieve a goal in a neighborhood of a certain point.

\vec{r}_i as in Equation 2.10:

$$\vec{f}_{\alpha i}(\|\vec{r}_{\alpha i}\|, t) := -\nabla_{\vec{r}_{\alpha i}} W_{\alpha i}(\|\vec{r}_{\alpha i}\|, t) \quad (2.10)$$

where $\vec{r}_{\alpha i} := \vec{r}_{\alpha} - \vec{r}_i$. For attractive effects, the attractiveness $\|\vec{f}_{\alpha i}\|$ decreases with time t since the interest descends. These types of forces are accountable for group formation.

To generalize these equations for attractive and repulsive effects for situations where not only the desired direction $\vec{e}_{\alpha}(t)$ of motion is taken into account, some adjustments must be made. In practice, context related to the behind of an agent influences less than what is in the front. Hence, this behavior is modeled by introducing a weaker influence factor c with $0 < c < 1$. To complete the perception about the effective angle 2φ of sight, forces must be weighted dependent on this angle of view:

$$w(\vec{e}, \vec{f}) := \begin{cases} 1 & \text{if } \vec{e} \cdot \vec{f} \geq \|\vec{f}\| \cdot \cos\varphi \\ c & \text{otherwise} \end{cases} \quad (2.11)$$

The repulsive and the attractive effects on an agent's behavior are formalized through Equations 2.12 and 2.13:

$$\vec{F}_{\alpha\beta}(\vec{e}_{\alpha}, \vec{r}_{\alpha} - \vec{r}_{\beta}) := w(\vec{e}_{\alpha}, -\vec{f}_{\alpha\beta}) \cdot \vec{f}_{\alpha\beta} \cdot (\vec{r}_{\alpha} - \vec{r}_{\beta}) \quad (2.12)$$

$$\vec{F}_{\alpha i}(\vec{e}_{\alpha}, \vec{r}_{\alpha} - \vec{r}_i, t) := w(\vec{e}_{\alpha}, \vec{f}_{\alpha i}) \cdot \vec{f}_{\alpha i} \cdot (\vec{r}_{\alpha} - \vec{r}_i, t) \quad (2.13)$$

Finally, pedestrians' total motion is modeled by summing all effects above mentioned since they all contribute at the same instant of time. It results in Equation 2.14:

$$\begin{aligned} \vec{F}_{\alpha}(t) &:= \vec{F}_{\alpha}^0(\vec{v}_{\alpha}, v_{\alpha}^0 \vec{e}_{\alpha}) + \sum_{\beta} \vec{F}_{\alpha\beta}(\vec{e}_{\alpha}, \vec{r}_{\alpha} - \vec{r}_{\beta}) \\ &+ \sum_B \vec{F}_{\alpha B}(\vec{e}_{\alpha}, \vec{r}_{\alpha} - \vec{r}_B^{\alpha}) + \sum_i \vec{F}_{\alpha i}(\vec{e}_{\alpha}, \vec{r}_{\alpha} - \vec{r}_i, t) \end{aligned} \quad (2.14)$$

Last, since the initial goal was to model the temporal changes of the preferred velocity $\vec{w}_{\alpha}(t)$ of a pedestrian α (e.g., in general, an agent), the Social Force Model is defined by Equation 2.15:

$$\frac{d\vec{w}_{\alpha}}{dt} := \vec{F}_{\alpha}(t) + \text{fluctuations} \quad (2.15)$$

where a fluctuation term has been added to describe random variations of motion from the nominal behavior.

The pedestrian dynamic model is completed by the relation between the goal velocity $\vec{w}_\alpha(t)$ and the current or actual velocity $\vec{v}_\alpha(t)$. By the fact that the current speed is limited by the maximum acceptable speed v_α^{max} which is the nominal value, the **realized motion** is:

$$\frac{d\vec{r}_\alpha}{dt} = \vec{v}_\alpha(t) := \vec{w}_\alpha(t) \cdot g\left(\frac{v_\alpha^{max}}{\|\vec{w}_\alpha\|}\right) \quad (2.16)$$

where

$$g\left(\frac{v_\alpha^{max}}{\|\vec{w}_\alpha\|}\right) := \begin{cases} 1 & \text{if } \|\vec{w}_\alpha\| \leq v_\alpha^{max} \\ v_\alpha^{max}/\|\vec{w}_\alpha\| & \text{otherwise} \end{cases} \quad (2.17)$$

As can be understood in the general formulation described above [38], despite the simplicity of this approach, the motion of pedestrians is taken into account enough realistically.

However, its performance decreases in difficult situations like very crowded environments. Indeed, in the next section, the approach developed in [48] uses a relatively small number of humans the robot has to deal with while navigating in a quite simple environment. Stochastic behavioral models may be developed if one restricts the description of behavioral probabilities that can be found in a huge population of individuals.

2.2.2 Waypoint-based path planner

In [48], the SFM-based local planner has been extended with the A^* algorithm to get effective social paths, like smoothed and socially compliant paths. The extension helps to avoid getting stuck in local minima and to take into account social zones for human comfort.

In practice, the proposed method in [48] produces smoothed paths that respect people's social space without unnecessary replanning increases a lot of performance indexes.

Since SFM has a flexible natural behavior by its mathematical formulation, authors in [48] extended it with social cues and signals. Taking into account that in a human-robot shared dynamic environment, future trajectory collisions must be predicted and that a certain distance from people and obstacles must be kept, the most remarkable extensions are the following:

- Passing and crossing behavior: the collision prediction force applies a circular force at a specific angle to the potential conflict point between a robot and a human that are moving toward each other. Then, it lets accelerate the nearest agent to that point, while the other slows down;
- Instant turns and oscillations: these problems can occur because of instant changes in force size and discontinuity at certain points. To solve them, when the robot moves in any direction, smoothing is applied to consecutive time stamp forces. They are decreased or increased by a specific step size when their magnitudes are bigger than a threshold, imposing continuity on steering;
- Identifying force frame: given the origin based on the robot's pose, the performing of vector operations of forces is handled by publishing a force frame transformation (Figure 2.15);

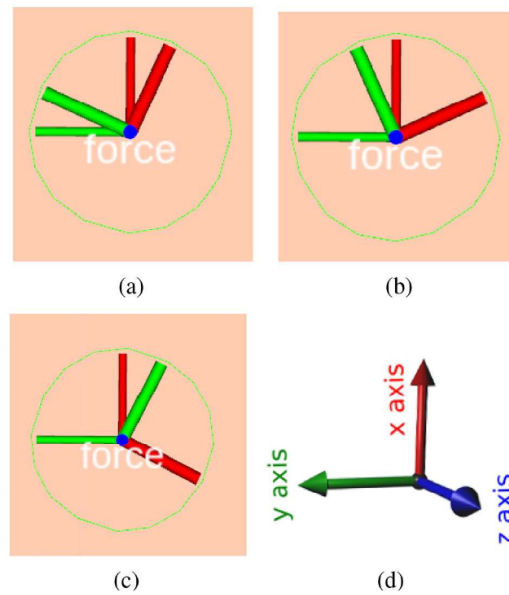


Figure 2.15: The force global frame is not changing while the robot is turning right in the approach proposed in [48].

- Robot controller: the output vector must be passed in input to the controller of the robot in polar coordinates (Figure 2.16). This is the reason why the total force vector is broken into x and y components: $f^{total} = f_{r\parallel\theta} + f_{r\perp\theta}$. Now, the robot acceleration can be written as $a_\omega = r \times f_{r\perp\theta} + k\omega$ or $a_\omega = k_p \cdot \Delta\alpha + k_d \cdot (-\omega)$. The linear velocities are $v_r = v_r + a_v \cdot \Delta t$ and $\omega_r = \omega_r + a_\omega \cdot \Delta t$. About rotation, it occurs as a yaw angle around the robot's

z-axis as in Equation 2.18:

$$\theta := \begin{cases} \theta - (2\pi) & \text{if } \pi < \theta < 2\pi \\ \theta + (2\pi) & \text{if } -2\pi < \theta < -\pi \end{cases} \quad (2.18)$$

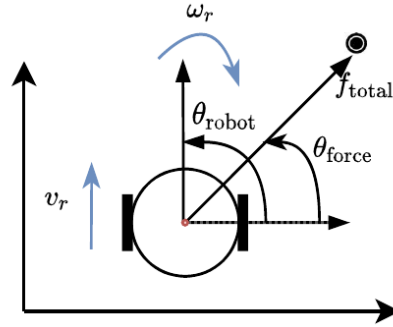


Figure 2.16: Robot's main features about coordinates, rotation velocities, and total SFM force [48].

- Avoiding local minima: when a robot gets stuck in a local minima, it never reaches the goal since attractive and repulsive forces balance out⁷ (Figure 2.17). This is solved by authors with a high-level global planner that finds a valid path between the starting point and goal point, to provide to the SFM-based local planner to execute the plan;

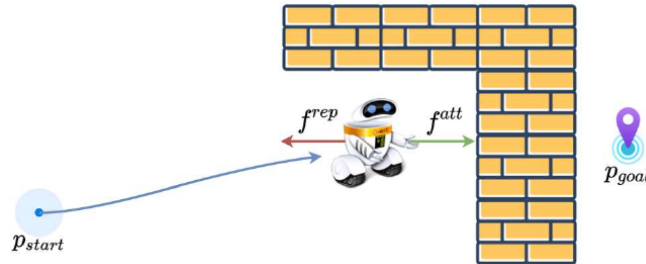


Figure 2.17: Example of local minima condition, where the sum of the attractive forces f^{att} and repulsive forces f^{rep} are balanced out. As a result, the robot gets stuck. The blue arrow shows the path followed by the robot to go from the starting point p^{start} to the local minima. The figure is taken from [48].

- Waypoints (social subgoals) selection algorithm: in dynamic and uncertain environments, re-planning is needed during plan execution since the plan tends to change frequently. To avoid this costly operation, Algorithm 3 [48] considers human social zones (costmaps) extracting waypoints by pruning

⁷The local minima problem happens when the direction of robot velocity, the obstacles and the goal position are on the same straight line that points from the robot center to the target location, passing through the obstacles.

parts of the global path and assigning them incrementally to the robot's path planner (Figure 2.19). It is based on the angle between the vectors of the nodes.

Algorithm 3 Waypoints (Social Subgoals) Selection

Input: Global Path $P = \{p_1, p_2, p_3, \dots, p_n\}$
Output: Waypoints $K = \{k_1, k_2, k_3, \dots, k_m\}$

- 1: Initialize K as an empty set;
- 2: **for all** $p_i \in P$ **do**
- 3: Calculate the angle value of node p_i :
- 4: $\vec{v}_1 \leftarrow \text{Compute}_{velocity}(p_i - p_{i-1})$
- 5: $\vec{v}_2 \leftarrow \text{Compute}_{velocity}(p_{i+1} - p_i)$
- 6: $\cos(\theta) \leftarrow \frac{(\vec{v}_1 \cdot \vec{v}_2)}{(\|\vec{v}_1\| \cdot \|\vec{v}_2\|)}$
- 7: **if** $\theta < \text{threshold}$ **then**
- 8: Erase p_i from the global plan P
- 9: **else**
- 10: Add p_i to waypoints K
- 11: **end if**
- 12: **end for**
- 13: **return** Waypoints K

The block diagram of the software architecture of the waypoint-based extended SFM planner is shown in Figure 2.18.

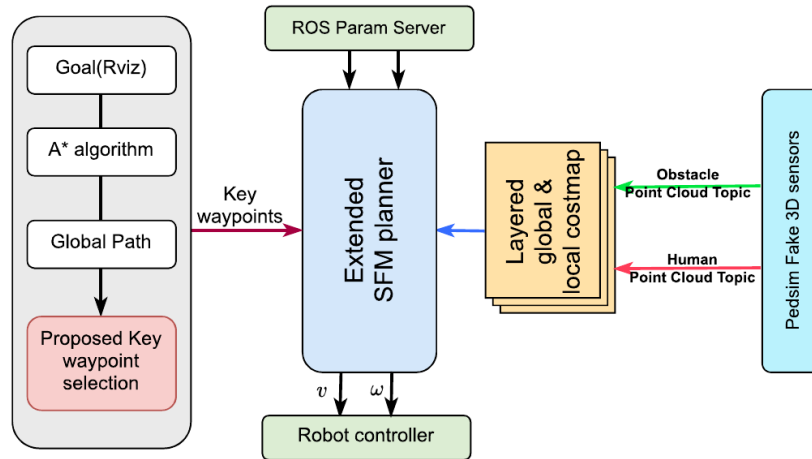


Figure 2.18: Software architecture overview of the waypoint-based extended SFM planner proposed in [48].

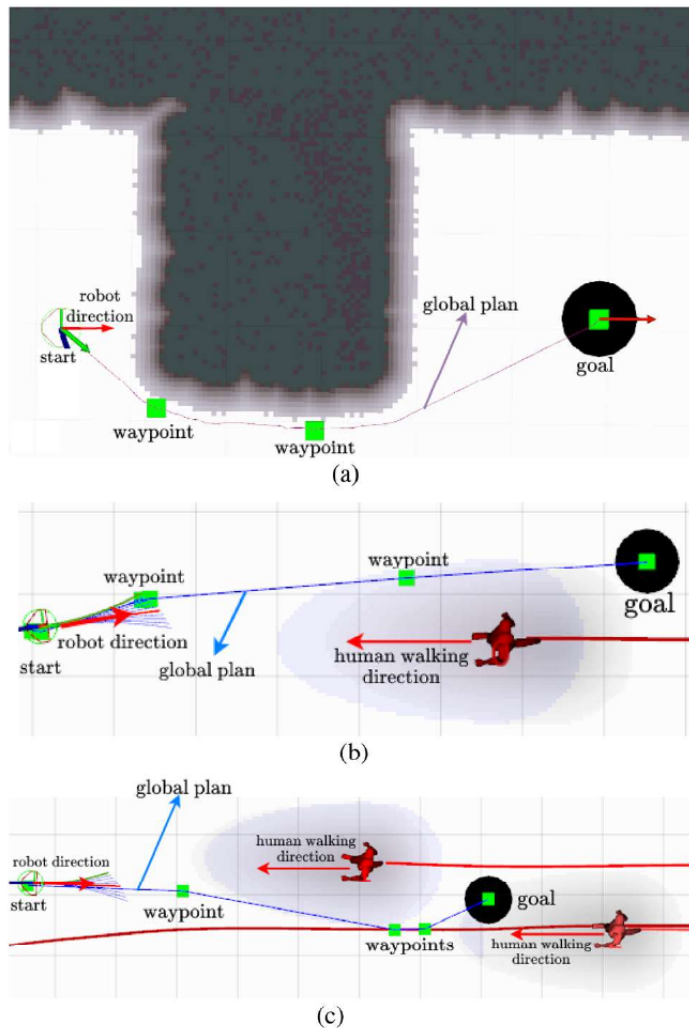


Figure 2.19: Process of finding or selecting and extracting waypoints on the global plan (solving local minima problem smoothly). The robot changes its global plan in all the three examples that are displayed. The waypoints have been added to avoid: (a) walls; (b) a man; (c) two men. Figure taken from [48].

Simulated Experiments and Results

Simulations for the work above ran in ROS Melodic on Ubuntu 18.04 using Pedestrian Simulator (PedSim) (Section 3.1.4). A corridor-like dynamic environment has been simulated (Figure 2.20) with possible scenarios where the robot could get stuck at the local minima. Comparing the traditional method with this approach, experimental results show the precedent method is inefficient, expensive in terms of time, and it is based on a rough path. Instead, running this method, human comfort, social acceptance, path smoothness, path length and performance indexes increase, avoiding most of the time problems like local minima one.

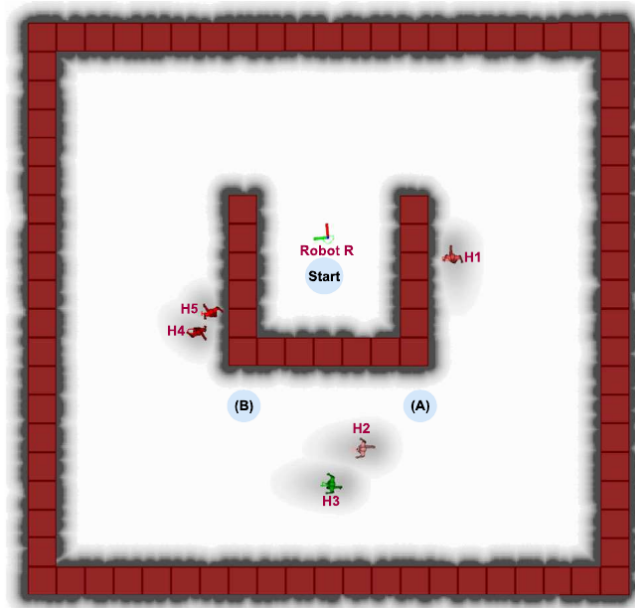


Figure 2.20: The corridor-like dynamic simulation environment consists of walls and moving humans where the robot can get stuck at the local minimum conditions. The robot R navigates consecutively from the start point (Start) through points (A) and (B), and then goes back to the start point while avoiding encountering dynamic humans H1-5 and obstacles in the environment. The figure is taken from [48].

More in detail, performances are reported in Table 2.1.

However, this method is neither tested in a real scenario nor a crowded scene, even simulated.

Evaluation metrics	Without waypoint approach	Proposed waypoint approach
Path execution time (s)	441	204
Path length (m)	43.652	40.030
# replanning	21	2
Total heading change ($^{\circ}$)	5094	2282

Table 2.1: Comparison of w/out waypoint selection approaches on the characteristic of path properties taken from [48].

2.3 Learning-based Approaches

In this section three important works in the application of learning methods in SARN are presented. They are based on different learning approaches (e.g., DNN, Encoder-Decoder, LSTM, RL, etc.), sometimes merged with other types of operations.

2.3.1 Human Trajectory Prediction in Crowds

In [51], the problem of human trajectory prediction has been cast as learning a representation of human social interactions. Current human trajectory prediction works can be categorized into learning human-human interactions or human-space interactions, more properly called social and physical interactions, respectively. The work in [51] belongs to the first category and it is focused on short-term human trajectory prediction (about the next five seconds).

In addition, the notion of primary pedestrian⁸ is crucial for a better understanding.

The problem of human trajectories prediction takes in input the trajectories of all people involved in a scene, where people are denoted by $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$ and respective future trajectories $\mathbf{Y} = \{Y_1, Y_2, \dots, Y_n\}$. At each time step t the pedestrian i is represented by $\mathbf{x}_i^t = (x_i^t, y_i^t)$, its velocity \mathbf{v}_i^t , and its state $\mathbf{s}_i^t = [\mathbf{x}_i^t, \mathbf{v}_i^t]$. From the positions of pedestrians in the time interval $t = \{1, \dots, T_{obs}\}$, future positions must be predicted in the time interval $t = \{T_{obs+1}, \dots, T_{pred}\}$. Predictions are labeled with $\hat{\mathbf{Y}}$.

For their work purposes, authors of [51] decided to categorize types of trajectories based on each scene concerning its corresponding primary pedestrian (Figure 2.21):

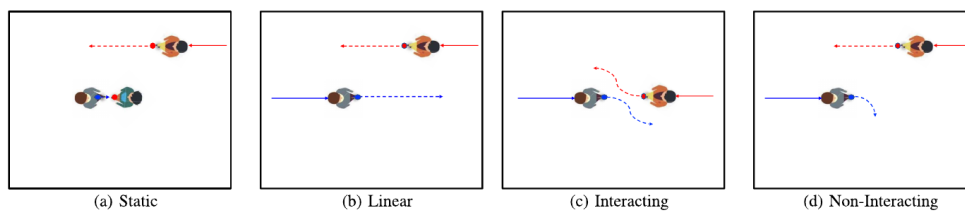


Figure 2.21: Visualization of the four high-level defined trajectory categories according to [51]. The category takes into account the shape of the trajectories and whether there will be or not an interaction between humans.

1. *Static*: the Euclidean displacement of the primary pedestrian in the scene with another human is less than a threshold (Figure 2.21(a));
2. *Linear*: the trajectory of the primary pedestrian can be correctly predicted with the help of an Extended Kalman Filter (EKF) ([75], [46]) (Figure 2.21(b));

⁸A primary pedestrian is a particular pedestrian of interest in the scene according to [51].

3. *Interacting*: called Type III trajectories (Figure 2.21(c)), these regard contexts in which the primary pedestrian experiences social interactions. They can be divided again into other four types as in Figure 2.22;

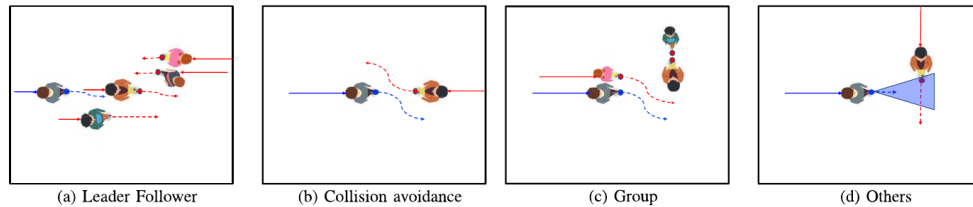


Figure 2.22: Visualization of Type III interactions commonly occurring in real-world crowds proposed in [51].

4. *Non-Interacting*: the trajectory of the primary pedestrian is non-linear and undergoes no social interactions during prediction (Figure 2.21(d)).

As introduced above, given a scene the goal is to predict all future people trajectories present in that scene. Figure 2.23 shows the global pipeline for predicting human motion. The main modules are the Motion Encoding Module, the Interaction Module, and the Decoder Module. Briefly, the first module encodes the past pedestrian’s motions, while the second learns to capture social interactions. The output is the social representation of the scene, passed in input to the third main module, which predicts a single trajectory or a trajectory distribution.

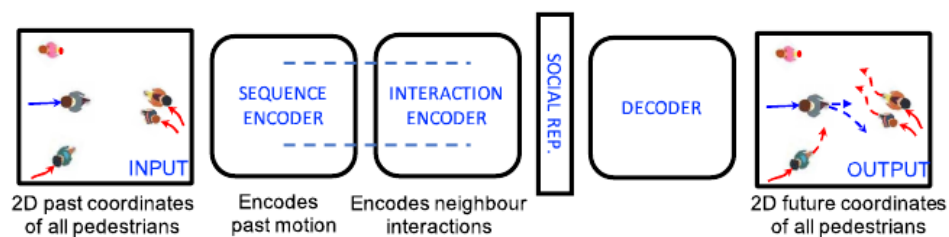


Figure 2.23: A data-driven pipeline for human trajectory prediction. The focus of the work proposed in [51] is on the design choices for the interaction module.

The choices in [51] take into account the different types of data-driven interaction encoders present in the literature. They can be divided into the following way:

- *Grid-based Interaction Models*: the interaction module gets input from a local grid built around the primary pedestrian. Each cell represents a particular spatial position relative to that human. The neighbor input state

representation brings different designs of models: Neighbor Input State with Occupancy Pooling (Figure 2.24 (a)), Social Pooling (Figure 2.24 (c)) or Directional Pooling (Figure 2.24 (b)), used in [51];

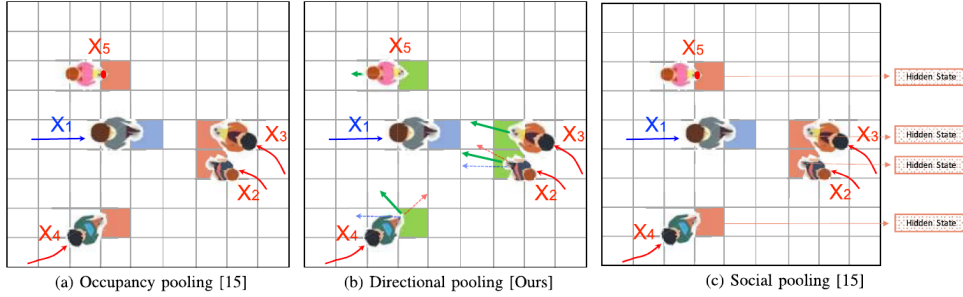


Figure 2.24: Illustration of the grid-based interaction encoding modules. (a) Occupancy pooling: each cell indicates the presence of a neighbor. (b) Directional pooling proposed in [51]: each cell contains the relative velocity of the neighbor concerning the primary pedestrian. (c) Social pooling: each cell contains the LSTM hidden state of the neighbor. The constructed grid tensors are passed through an MLP-based neural network to obtain the interaction vector. Figure taken from [51].

- *Non-Grid-based Interaction Modules:* they capture social interactions in a grid-free way. They are influenced by four factors (Figure 2.25): Neighbor Input State, Input State Embedding, Aggregation strategy, and Aggregated Vector Embedding.

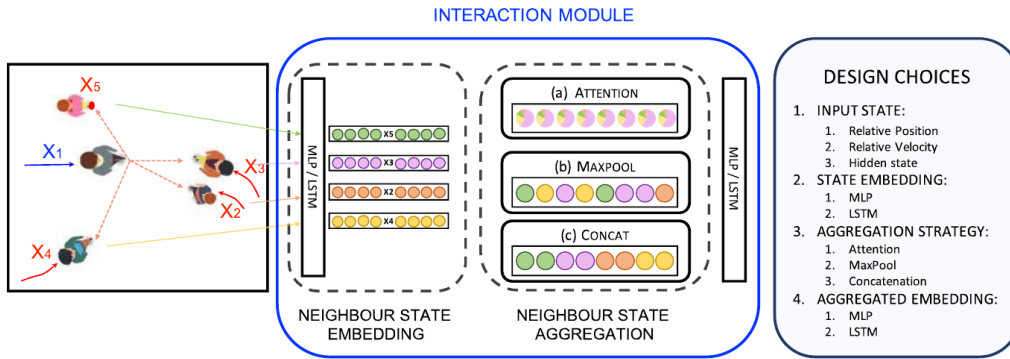


Figure 2.25: Illustration of the non-grid-based encoding modules to obtain the interaction vector. The challenge lies in handling a variable number of neighbors and aggregating their state information to construct the interaction vector. (a) Neighbor information is aggregated via attention mechanism. (b) Neighbor information is aggregated utilizing a symmetric function. (c) Neighbor information is aggregated via concatenation. Figure taken from [51].

To build the prediction model, the time-sequence encoder has been chosen to be an LSTM since it can handle and capture long-term dependencies. The state \mathbf{s}_i^\dagger is embedded using a single-layer Multi-Layer Perceptron (MLP) to get the state embedding \mathbf{e}_i^\dagger . It follows that the interaction vector \mathbf{p}_i^\dagger of person i is

computed by the interaction encoder. Now, the concatenation of the interaction vector with the velocity embedding is the input to the sequence encoding module. Finally, the hidden state of the LSTM at time step t of pedestrian i is the one used to predict the distribution of the velocity at time step $t + 1$.

The method used to develop this work can be defined to be a Layerwise Relevance Propagation (LRP) [62] method. This type of work is one of the most prominent in ML. LRP distributes the decision outputs of the model back to each of the input variables. This suggests each input contributes to the output. Furthermore, LRP affects which neighbors and past velocities of the primary pedestrian the model is focusing on during the regression of the next velocity prediction. Shortly, LRP techniques are generic and applicable as bases of any trajectory prediction network.

Tests performed in [51] showed the proposed models outperformed competitive baselines on a synthetic dataset specially built. Instead, on the real dataset there was no clear winner against all designs in terms of distance metrics and other indexes. However, a notable reduction of model prediction collisions has been gained. Also, the quality of trajectory prediction models of the work proposed in [51] can be still increased.

2.3.2 Genetic algorithm for learning to plan people-aware trajectories

The work in [9] presents a computationally light method to make a robot learn to implement social behaviors according to proxemic rules, without worsening traditional navigation performances. The method is based on a genetic algorithm to solve the motion planning task problem by optimizing the obstacle avoidance, path length, and the computation of social trajectories. In this way, the parameters of traditional local planners are automatically tuned. Everything is handled by exploiting an RL technique, for which the robot is trained to perform the navigation task by attempting to maximize the reward function.

The pipeline is represented in Figure 2.26 and it is inspired by the Genetic Algorithm (GA), the method searches for the optimal solution in the space spanned by the chromosomes. The optimization problem is made up of a sequence of variables, called genes, that represent chromosomes. Then, starting from the initial population, some steps lead the algorithm to convergence in expectation of a near-optimal solution. These steps are reproduction, selection, crossover, and mutations, in cycles. This is the main principle formalized in Algorithm 4 to opti-

mize the set of parameters related to path planning provided by ROS navigation stack⁹.

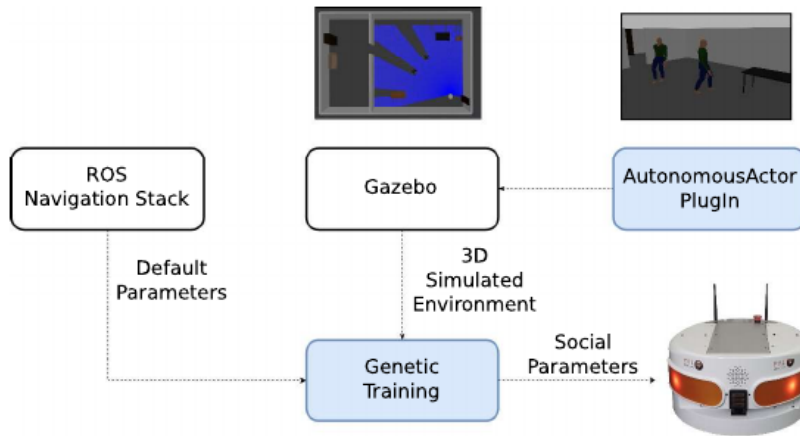


Figure 2.26: The pipeline of the work proposed in [9]. The blue blocks represent the main contributions of [9], while the others represent software that authors of [9] only used.

Moreover, each chromosome gets a score depending on the following three indices:

- Time score s_{time} ;
- Social score $s_{distance}$;
- Social score s_{social} .

The final score function is defined as in Equation 2.19, where w_{st} , w_{ss} and w_{sd} are defined to highlight the s_{social} .

$$s_{chromo} = w_{st} \cdot s_{time} + w_{ss} \cdot s_{social} + w_{sd} \cdot s_{distance} \quad (2.19)$$

To update the current population in each cycle, reproduction, mutation, and selection operators have been specifically defined by the authors [9].

Simulations, training, and testing phases proposed in the work [9] ran on a computer equipped with Intel Core i5-6660 Central Processing Unit (CPU) and 16 GigaByte (GB) Random Access Memory (RAM). Based on virtual forces generated on the fly when a probable collision is detected, the obstacle avoidance functionality improves.

Experiments done in [9] used the AutonomousActorPlugIn¹⁰, which is a GAZEBO (Section 3.1.1) plug-in¹¹ to manage the 3D human models of actor dynamics. The

⁹For further details: <https://wiki.ros.org/navigation>.

¹⁰For further details: <https://github.com/bach05/gazebo-plugin-autonomous-actor>.

¹¹For further details: https://gazebosim.org/tutorials?tut=actor&cat=build_robot.

Algorithm 4 Genetic training

Input: $gen, pop, seed_chromo, \sigma^2$

```

1: for  $i \leftarrow 1$  to  $pop$  do
2:   population  $\leftarrow seed\_chromo + \mathcal{N}(0, \sigma_1^2)$ ;
3: end for
4: for  $k \leftarrow 1$  to  $gen$  do
5:   //Evaluation
6:   for all  $chromo \in population$  do
7:     computeScore( $chromo$ );
8:   end for
9:   count  $\leftarrow$  countWorstChromos(population);
10:  while  $size(population) < pop + count$  do
11:    Pick parents  $\in population$  randomly;
12:    //Reproduction + //Mutation
13:    population  $\leftarrow$  newChromo(parents) +  $\mathcal{N}(0, \sigma_2^2)$  with  $prob$ ;
14:  end while
15:  //Selection
16:  removeWorsts(population, count);
17: end for

```

robot is the robot platform TIAGo base¹² (Section 3.2). Experiments have been conducted in a simulated office and a simulated house. Both training of the genetic model and its testing have been done with and without the use of Proxemic social_navigation_layers¹³ of ROS to alter the costmaps around pedestrians positions. After the training of the 36 chromosomes for 24 generations and a probability of mutation of 50%, testing navigation accuracy achieved is about 70-75% in both environments, in all conditions. It means the approach of [9] does not disturb the traditional navigation performances, proving in this way the reliability and robustness of the system.

When the social layer is not included the global path does not respect social rules and dynamic obstacles are sometimes in collision, while with it the right behaviour is shown, as expected. Furthermore, [9] proves with its simple optimization algorithm it is possible to obtain good performances using an extension of the social layers, without using other tools too expensive and heavy. Still, the work can be extended to capture more complex social behaviors about Human-Robot Interaction (HRI) and, possibly, increase a little bit performance again.

¹²For further details: <https://pal-robotics.com/robots/tiago-base/> .

¹³For further details: https://wiki.ros.org/social_navigation_layers .

2.3.3 Neuro-Symbolic Approach for Enhanced Human Motion Prediction

In [61], the scope is on a new approach for context reasoning in the research field of human motion prediction. The authors propose a neuro-symbolic approach, called NeuroSyM, that can weigh the different interactions in the neighborhood by exploiting the spatial representation technique of Qualitative Trajectory Calculus (QTC) [36].

In [61] the motion prediction is based on weighted interactions embedding between pairs of agents. Starting from raw trajectories as inputs, the following two architectures have been chosen:

- *NeuroSyM SGAN*: it is a type of Social Generative Adversarial Network (SGAN) famous in literature [35] because it can increase accuracy and speed of human motion prediction in crowds. The performance indices for the model proposed in [61] (Figure 2.27) are ADE and FDE, respectively defined in Equations 2.20 and 2.21, where T_{pred} is equal to the number of steps predicted and each step measures 0.4 seconds, N is the total number of training trajectories, \tilde{X} and X are respectively the predicted trajectory and the true trajectory;

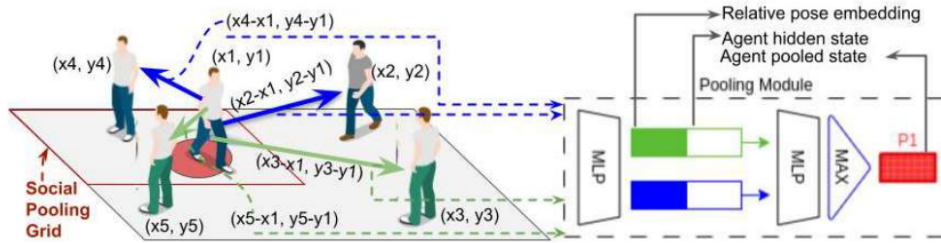


Figure 2.27: The neuro-symbolic SGAN pooling mechanism proposed in [61]. The difference with the original SGAN pooling mechanism can be seen from the mixed arrows color added within and outside the red grid to represent different types of spatial relations or interactions with the central agent standing on the red spot, which can be inferred from the NeuroSyM SGAN architecture according to [61].

$$ADE = \frac{\sum_{i=1}^N \sum_{t=1}^{T_{pred}} \|\tilde{X}_t^i - X_t^i\|^2}{N * T_{pred}} \quad (2.20)$$

$$FDE = \frac{\sum_{i=1}^N \|\tilde{X}_{T_{pred}}^i - X_{T_{pred}}^i\|^2}{N} \quad (2.21)$$

- *NeuroSyM DA-RNN*: it is a Dual-stage Attention (DA) Recurrent Neural network (RNN) prediction architecture that does not use a pooling mech-

anism to overcome the size problem of dynamic input series. Indeed, it weights each single input data by assigning the proper attention to each one separately. In literature, it seems a good choice for time-series predicting [70]. Figure 2.28 shows where the NeuroSyM module acts on the original DA-RNN model with the introduction of a Conceptual Neighbourhood Diagram (CND) layer between the embedding and the softmax layers.

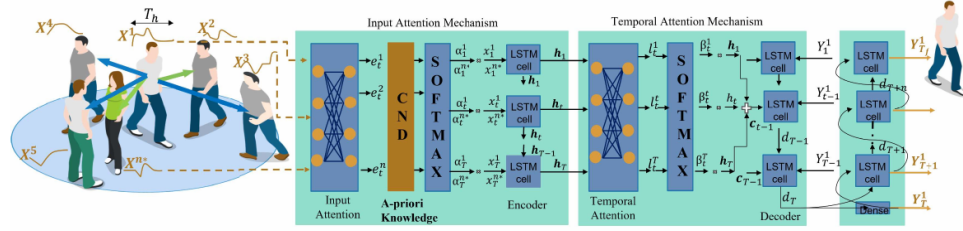


Figure 2.28: A neuro-symbolic approach for attention-based time-series prediction models proposed in [61]. Differently from SGAN-like architectures, attention-based mechanism have no pooling modules. The diagram is extended from [69] and modified for multi-step attention-based context-aware human motion prediction in crowded environments. The input are n^* time series of agents, within a cluster centered at the first time series, while the output is the prediction of the cluster center's agent. The vector \mathbf{e} denotes the input embeddings normalized to α after passing through the CND layer, which adds a-priori knowledge to them in the form of $\alpha_{cnd,t}^k$. The CND layer weights differently the spatial relations (represented by mixed arrows color) of the neighbor agents with the central one. The vector \mathbf{I} denotes the temporal attention weights of the encoder's hidden states output, normalized to β , while \mathbf{c} represents the context. $\mathbf{X} = \{x, y\}$ is the input driving vector; $\mathbf{Y} = \{x', y'\}$ is the label vector; \mathbf{h} and \mathbf{d} are the encoder and the decoder hidden states, respectively. The input and temporal attention layers are constructed from dense layers according to [61].

The encoder attention weights, called *alpha* in Figure 2.28, measure the importance of each input series at time t on the output prediction at $t + 1$. In Equation 2.22 $\exp(e_t^k)$ is the embedding of the k^{th} input series at time t (Equation 2.23), \mathbf{h}_{t-1} and \mathbf{s}_{t-1} are the hidden and cell state of the encoder LSTM at the previous step.

$$\alpha_t^k = \frac{\exp(e_t^k)}{\sum_{i=1}^n \exp(e_t^i)} \quad (2.22)$$

$$e_t^k = \text{dense} \left[\tanh \left(\text{dense}(\mathbf{h}_{t-1}; \mathbf{s}_{t-1}) + \text{dense}(\mathbf{x}_{1..T_h}^k) \right) \right] \quad (2.23)$$

Overall, experimental results in [61] show NeuroSyM outperforms in most cases the standard architectures about prediction accuracy. Precisely, tests focused on medium and long-term time horizons using two famous architectures in this field and six datasets have been used to challenge crowded scenarios tests.

Simulations results show that:

- *NeuroSyM SGAN*: the average relative gains for ADE and FDE over the datasets for $T_{pred} = 8, 12$ time steps are collected in Table 2.2;

Measure	Average Relative Gain (%)
ADE	+60.84 / +78.58
FDE	+58.40 / +76.97

Table 2.2: Performance comparison between the average relative gain across all datasets considered. The results' format refers to the 8/12 prediction time steps. ADE and FDE values are in meters. The results are taken from [61].

- *NeuroSyM DA-RNN*: evaluated over medium and long time horizons (48 and 80 time steps, respectively), Table 2.3¹⁴ reports the indices Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) computed between the predicted (x', y') coordinates and the true labels (x, y) .

Architecture	RMSE	MAE
DA-RNN (Baseline)	3.61 / 3.572	2.097 / 2.753
NeuroSyM DA-RNN	2.815 / 3.728	2.162 / 2.166
Relative Gain (%)	+22 / -4.37	-3.1 / +21.32

Table 2.3: Performance comparison between the baseline architecture DA-RNN and the NeuroSyM approach on the JackRabbit dataset. The results' format refers to the 48/80 prediction time steps. RMSE and MAE values are in meters, and the best results are highlighted in bold (i.e. the lower error, the better). The results are taken from [61].

2.4 Active Perception

This section presents an overview about the work of active perception by focusing on the contribution of each mentioned work.

2.4.1 Classical Approaches

About the application of classical approaches to the topics of social navigation and people detection, Table 2.3 lists a considerable collection of papers and highlights the features of each approach:

- The single or multi-pedestrian is related to the presence of one or more pedestrians or humans involved in the study, and to the pedestrian path prediction whether implemented;

¹⁴For further details on JackRabbit: <https://svl.stanford.edu/projects/jackrabbit/>

- The group awareness whether groups of people are properly taken into account and whether crowds are the main theme of contexts considered;
- The presence of links to a code repository;
- The specification of the metrics chosen as performances evaluation indexes;
- The display of results of simulation tests and real-world tests.

In short, geometrical approaches have been widely employed in SARN to model and reason about the robot's environment. These methods leverage geometric representations and algorithms to process and manipulate simple data to geometric representations, such as occupancy grids or potential fields, to plan robot trajectories and avoid collisions with obstacles and humans. While geometrical methods offer simplicity and efficiency in planning robot motion, they often struggle with handling complex social interactions. They often assume static environments and can not capture higher-level social cues and intentions. Furthermore, these methods are prone to local minima and can result in unnatural or inefficient robot behavior. To handle these limitations and to enable more efficient and accurate solutions, researchers have turned to SFMs and learning-based approaches.

SFM methods have gained significant attention in recent years due to their ability to model and simulate complex crowd dynamics. Inspired by the principles of social psychology, these methods employ forces to represent interactions among individuals within a crowd. By considering factors such as social norms, personal preferences, environmental constraints, goals, and the forces exerted by other agents, SFMs can predict and simulate crowd behaviors with enough accuracy. However, accurately calibrating the parameters of these models can be challenging, as they often require empirical data and extensive parameter tuning. In addition, these methods may not fully capture the complex cognitive processes involved in human decision-making. Researchers continue to enhance SFM by incorporating ML techniques and real-time data analysis, enabling more accurate predictions and more effective crowd-control strategies. However, the choice of learning methods is constantly growing since they have revolutionized the field of computer engineering. These methods strengthen the power of data-driven models to automatically learn patterns and make predictions or decisions.

Learning methods, particularly those based on ML and DNN, have acquired significant attention in recent years for their ability to capture a lot of complex

patterns and behaviors in social navigation. These methods can learn from large-scale datasets to infer social norms, predict human motion and generate robot trajectories that adhere to social conventions. RL algorithms have been employed to optimize robot behavior by rewarding socially acceptable actions. One notable advantage of learning methods is their ability to adapt and generalize. By training on diverse datasets, these models can handle a wide range of social scenarios and variations in human behavior. However, learning-based approaches often require substantial amounts of annotated data, which can be costly and time-consuming to acquire. Furthermore, it is very hard to determine how to tune the parameters to achieve the desired robot’s behavior, mainly in social contexts, resulting in tedious and time-consuming tuning. Unfortunately, performances are strongly dependent on a lot of parameters that the user should set according to the specific situation. Additionally, the outputs of learning-based approaches may not be interpreted by humans, making it challenging to understand why a particular decision or action was taken by the robot.

Finally, the absence of a singularly superior method often leads to the common choice to mix different approaches for enhanced performances.

Title	Work	Single/Multi pedestrian	Pedestrians prediction	Group Awareness	Crowds	With code	Evaluation metrics	Simulation tests	Real world tests
Uncertainty-aware Navigation in Crowded Environment	[3]	M	×	×	✓	×	✓	✓	×
Developing a VR Socially Assistive Robot Simulator Employing Game Development Tools	[4]	M	×	×	×	✓	×	×	×
Learning to plan people-aware trajectories for robot navigation: A genetic algorithm	[9]	M	✓	×	×	✓	×	✓	×
SG-LSTM: Social Group LSTM for Robot Navigation Through Dense Crowds	[15]	M	✓	✓	✓	×	✓	✓	✓
Multi-agent navigation in human-shared environments: A safe and socially-aware approach	[16]	M	✓	×	×	×	✓	✓	✓

Title	Work	Single/Multi pedestrian	Pedestrians prediction	Group Awareness	Crowds	With code	Evaluation metrics	Simulation tests	Real world tests
Human-Aware Path Planning With Improved Virtual Doppler Method in Highly Dynamic Environments	[17]	M	×	✓	✓	×	✓	✓	×
Autonomous Navigation by Mobile Robots in Human Environments: A Survey	[18]	S	✓	×	×	×	✓	×	×
Detecting Anomalous Behaviour of Socially Assistive Robots in Geriatric Care Facilities	[19]	S	×	×	×	×	×	×	×
Towards using Behaviour Trees for long-term social robot behaviour	[20]	S	×	×	×	✓	×	×	×
Human Re-Identification with a Robot Thermal Camera Using Entropy-Based Sampling	[22]	M	×	×	×	✓	✓	×	✓
Learning World Transition Model for Socially Aware Robot Navigation	[24]	M	✓	×	×	✓	✓	✓	✓
FlowBot: Flow-based Modeling for Robot Navigation	[25]	M	✓	×	✓	×	✓	✓	×
Using Socially Assistive Human-Robot Interaction to Motivate Physical Exercise for Older Adults	[26]	S	×	×	×	×	✓	×	✓
Robot Companion: A Social-Force based approach with Human Awareness-Navigation in Crowded Environments	[28]	M	✓	✓	✓	×	✓	✓	✓
Social Robot Navigation Tasks: Combining Machine Learning Techniques and Social Force Model	[32]	M	✓	×	×	×	✓	✓	✓
SocialGym: A Framework for Benchmarking Social Robot Navigation	[39]	M	×	✓	×	×	✓	✓	×
Group Estimation for Social Robot Navigation in Crowded Environments	[47]	M	✓	✓	✓	×	✓	✓	×
Waypoint based path planner for socially aware robot navigation	[48]	M	✓	×	×	✓	✓	✓	×
Human Trajectory Forecasting in Crowds: A Deep Learning Perspective	[51]	M	✓	✓	×	×	✓	✓	×

Title	Work	Single/Multi pedestrian	Pedestrians prediction	Group Awareness	Crowds	With code	Evaluation metrics	Simulation tests	Real world tests
Socially aware robot navigation framework in crowded and dynamic environments: A comparison of motion planning techniques	[53]	M	×	×	✓	×	✓	✓	×
Mobile Robot Navigation Using Deep Reinforcement Learning	[54]	S	×	×	×	×	✓	✓	✓
Exploring the Effect of Mass Customization on User Acceptance of Socially Assistive Robots (SARs)	[57]	M	×	×	×	×	✓	×	✓
Analysis of Tiago Robot for Autonomous Navigation Applications	[60]	M	×	×	×	×	✓	✓	×
A Neuro-Symbolic Approach for Enhanced Human Motion Prediction	[61]	M	✓	✓	✓	✓	✓	✓	×
Automatic Waypoint Generation to Improve Robot Navigation Through Narrow Spaces	[63]	S	×	×	×	×	✓	✓	✓
On equitably approaching and joining a group of interacting humans	[64]	M	×	✓	×	×	✓	✓	×
Pedestrian-Robot Interactions on Autonomous Crowd Navigation: Reactive Control Methods and Evaluation Metrics	[66]	M	✓	×	✓	✓	✓	✓	✓
ROS Based Heterogeneous Multiple Robots Control Using High Level User Instructions	[71]	S	×	×	×	×	✓	✓	×
Adaptive Side-by-Side Social Robot Navigation to Approach and Interact with People	[74]	M	✓	✓	✓	×	✓	✓	✓
Socially-Aware Navigation Planner Using Models of Human-Human Interaction	[76]	S	✓	×	×	×	✓	✓	×
Implementation of SARL* Algorithm for A Differential Drive Robot in a Gazebo Crowded Simulation Environment	[77]	M	✓	×	×	×	✓	✓	×
Group-Aware Human Trajectory Prediction	[78]	M	✓	✓	×	×	✓	✓	×

Title	Work	Single/Multi pedestrian	Pedestrians prediction	Group Awareness	Crowds	With code	Evaluation metrics	Simulation tests	Real world tests
Human-Aware Navigation Planner for Diverse Human-Robot Interaction Contexts	[79]	M	✓	×	✓	✓	✓	✓	✓
HATEB-2: Reactive Planning and Decision making in Human-Robot Co-navigation	[83]	S	✓	×	×	✓	✓	✓	✓
“To Approach Humans?”: A Unified Framework for Approaching Pose Prediction and Socially Aware Robot Navigation	[85]	M	✓	✓	×	×	✓	✓	✓
Dynamic Social Zone based Mobile Robot Navigation for Human Comfortable Safety in Social Environments	[84]	M	✓	✓	×	×	✓	✓	✓
RGB-D and Laser Data Fusion-based Detection and Tracking for Socially Aware Robot navigation Framework	[86]	M	✓	✓	×	×	✓	×	✓
Group and Socially Aware Multi-Agent Reinforcement Learning	[87]	M	×	✓	×	×	✓	✓	×
Modeling Cooperative Navigation in Dense Human Crowds	[88]	M	✓	✓	✓	×	✓	✓	×
Crowd-Aware Robot Navigation for Pedestrian with Pedestrians with Multiple Collision Avoidance Strategies via Map-based Deep Reinforcement Learning	[91]	M	✓	×	✓	✓	✓	✓	✓
Macroscopic and microscopic dynamics of a pedestrian cross-flow: Part I, experimental analysis	[93]	M	✓	×	✓	×	✓	×	✓
Macroscopic and microscopic dynamics of a pedestrian cross-flow: Part II, modelling	[94]	M	✓	×	✓	×	✓	×	✓

Table 2.3: Analysis of the state-of-the-art approaches based on the following parameters: (a) Single or Multi pedestrian; (b) Pedestrians prediction; (c) Group awareness; (d) Crowds; (e) Link to code; (f) Evaluation metrics; (g) Simulation tests; (h) Real word tests. If a work takes into account single or multi pedestrians, then symbols M or S are respectively added. For the other features, a tick or a cross will be added according to the study of that feature in the work.

2.4.2 Active Perception

The works described previously highlight some limitations in term of robot's perception. One effective way to limit these limitations can be Active Perception.

In the field of Active Perception there are still open challenges and a lot of work to do. Indeed, most recent works on this topic do not rely mainly on Active Perception behaviors, but they are based on more classical solutions like people detection, tracking or following (e.g., [55]), or on the use of vision exploiting cameras and sensors data. Crowded situations are not traditionally included since, except for a few works (e.g., [33], [58]), groups considered are at most made up of between 10 and 15 people (e.g., [72], [6], [89]). Even though these numbers of people in contexts may not be labeled as crowded, this is a subjective analysis based on the space available in the environment¹⁵. Some other approaches merge data incoming from more external sensors (e.g., [1], [31], [33], [56]) or refine computations with the help of Deep Learning (DL) predictions (e.g., [2], [6], [11], [42], [45]), cameras and vision systems (e.g., [2], [21], [23], [34], [42], [82], [89], [90]).

Table 2.3 collects interesting papers related to the topic of active perception and analyses the differences highlighting the gap between their work and the purpose of this thesis.

Title	Work	Gap
Tracking of Multiple People in Crowds Using Laser Range Scanners	[1]	Required use of external laser sensors. Crowd is considered.
Detection and tracking using 2D laser range finders and deep learning	[2]	Required use of camera and vision. No crowd. Use of DL.
Friendly Robot Outdoor Guide (FROG): A new people detection dataset for knee-high 2D range finders	[6]	Very recent approach (June 2023). Use of DL and dataset proposed. No crowd, but until 10/12 people.
Multi-modal active perception for information gathering in science missions	[7]	Useful for spatial environments.
Revisiting active Perception	[10]	Only theoretical general introduction.

¹⁵A given number of people in a small environment can be identified as a crowd, while the same people in a large environment can be placed in a very sparse way.

Title	Work	Gap
Semantic-aware Active Perception for Unmanned Aerial Vehicles (UAVs) using Deep Reinforcement Learning	[11]	For UAVs. Required use of camera and vision. Only regions in images. No crowd.
Online planning for multi-robot active perception with self-organising maps	[13]	Multi-goal path planning for active perception and data collection tasks. Required collection of 3D point clouds. No crowd.
Deep Person Detection in 2D Range Data	[14]	Less annotated dataset with respect to [6].
Active Visual Perception for Mobile Robot Localization	[21]	Required use of camera and vision. Intense use of landmarks for pose estimation. No crowd.
Socially-Aware Multi-Agent Following with 2D Laser Scans via Deep Reinforcement Learning and Potential Field	[23]	No properly crowd considered. Simple environment.
Vision and Radio Frequency Identification (RFID) data fusion for tracking people in crowds by a mobile robot	[31]	Required use of camera for vision data fusion. Used for human following tasks. Not properly crowd.
Simultaneous People Tracking and Localization for social Robots Using External Laser range Finders	[33]	Required use of a big number of external laser sensors. Crowd is considered.
Tracking People in a Mobile Robot From 2D LiDAR Scans Using Full Convolutional Neural Networks for Security in Cluttered Environments	[34]	No crowd. Very simple environment.
Self-Supervised Person Detection in 2D Range Data using a Calibrated Camera	[42]	Required use of camera for auto generation of labels. No crowd.
2DLaserNet: A deep learning architecture on 2D laser scans for semantic classification of mobile robot locations	[45]	Classification of locations using DL and 2D laser scan detection. This can be simplified starting from the map of the environment.
Person Tracking and Following with 2D Laser Scanners	[55]	People following tasks. Approach tested on different robot platforms. No crowd.

Title	Work	Gap
An Ensemble Learning Method for Robot Electronic Nose with Active Perception	[56]	Required use of custom sensors as the electronic nose.
Active Simultaneous Localization And Mapping (SLAM) in Crowded Environments	[58]	Only static environments considered. A lot of assumptions about the type of environment. A couple of crowds.
People2D: realtime people detection in 2D range data	[72]	Ready-to-use software to be tested and evaluated. No crowds, but some people.
Active Perception for Foreground Segmentation: An RGB-D Data-Based Background Modeling Method	[82]	Required use of camera and vision. No crowd.
Multiscale Adaptive-switch Random Forest (MARF): Multiscale Adaptive-Switch Random Forest for Leg Detection With 2-D Laser Scanners	[89]	Complex approach. No crowd, up to 5 people considered in one environment.
LiDAR and Camera Detection Fusion in a Real-Time Industrial Multi-Sensor Collision Avoidance System	[90]	Required use of camera and vision. No crowd.

Table 2.3: Collection of approaches where title and work define the analyzed paper. In the Gap column the limitations of such approaches are reported.

Chapter 3

Tools and methods

In this chapter, the tools and the methods used to develop the work of this thesis are provided: from the ROS framework, GAZEBO, RViz and PedSim simulators, with other useful packages like Distance Robust SPatial Attention and Auto-regressive Model (DR-SPAAM) detector, to the TIAGo robot that is the main character in this project.

3.1 Robot Operating System (ROS)

ROS¹ (Figure 3.1) is an open-source software framework for robots. It provides the services one would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides libraries and tools for writing robot software [92]. ROS aims to simplify robot complex tasks, encouraging collaborative robotics software development. It enables efficient file-handling systems, hardware abstraction, and package management.



Figure 3.1: ROS icon.

Since several languages like C/C++ and Python² can be used to build software with the ROS platform, code across multiple computers can be built, written and ran. ROS communicates between its files and simulation software through

¹For further details: <https://www.ros.org/> .

²For further details: <https://www.python.org/> .

nodes. ROS uses the topic that will be publishing, subscribing, or doing both. ROS can be considered the backbone of modern robotics systems for its easiness in debugging and manipulating code.

The primary goal of ROS is to support code reuse in robotics research and development. Following this idea, developers can reuse code from other universities and improve it with their proper research. This approach contributes directly to science, where state-of-art techniques can be improved by sharing the code.

ROS currently runs on Unix-base platforms. The work developed for this thesis has exploited the ROS Melodic version with Ubuntu 18.04.6 Long-Term Support (LTS) bionic. The Linux kernel version is 5.4.0-84-generic. The distribution is compatible with TIAGo (Section 3.2) robot by PAL Robotics.

3.1.1 GAZEBO

GAZEBO³ (Figure 3.2) is an open-source well-designed simulator that makes possible test robot task solutions using realistic scenarios. These scenarios can be modeled to be similar to real-world indoor and outdoor environments through the specification of world model files. It provides a robust physics engine and a graphical interface to help the test phase and the simulation of the environment.



Figure 3.2: GAZEBO simulator icon.

This open-source simulation environment works with the following main components:

- 3D Unified Robot Description Format (URDF) models;
- Controllers;
- Tutorials.

More specifically, the architecture of the GAZEBO is a Client/Server architecture. It uses the publishers and subscribers for their inter-process communication. This simulator has a standard Player interface and additionally, it has a native interface.

³For further details: <https://staging.gazebo.org/home> .

In the process of dynamic simulation, GAZEBO can access the high-performance physics engines like Open Dynamics Engine (ODE), Bullet, Simbody, and Dynamic Animation and Robotics Toolkit (DART) which are used for rigid body physical simulation. Object-Oriented Graphics Rendering Engine (OGRE) provides the 3D graphics rendering of environments of GAZEBO.

The PAL Robotics team provides TIAGo simulation using ROS and GAZEBO. ROS plugins help in implementing a direct communication interface from GAZEBO to ROS. The TIAGo simulation model allows a smooth transition from simulation to the robot. Thereby both the real world and the simulated robot are controlled using the same software.

Simulated experiments explained in this thesis have been carried out also in this simulation environment.

3.1.2 ROS Visualization (RViz)

RViz⁴ (Figure 3.3) is a powerful 3D visualization tool for ROS. It allows users to view the simulated robot model, log sensor information from the robot's sensors, and replay the logged sensor information. In other words, it shows the robot's perception of its world, whether real or simulated.



Figure 3.3: RViz simulator icon.

RViz handles the visualization with a visualizer in which users can add, modify, remove, and rename displays. A display can draw some objects in the 3D world, and likely it has some options available in its displays list.

Different configurations of displays are often useful for different uses of the visualizer. To this end, the visualizer allows users to load and save different configurations. Moreover, different camera types can be used, and a lot of other

⁴For further details: <https://wiki.ros.org/rviz>.

tools can be exploited, just as coordinate frames, goal navigation, pose estimation, and time.

Since RViz shows the robot's perception, it can work well also with TIAGo by PAL Robotics without meaningful problems.

Simulated experiments explained in this thesis have been carried out also in this simulation environment.

3.1.3 Distance Robust SPatial Attention and Auto-regressive Model (DR-SPAAM)

DR-SPAAM⁵ (Figure 3.4) is a ROS package for 2D person detection in 2D range data based on the work in [41]. It is a package useful for detecting people standing or moving using only data incoming from 2D LiDAR data and for generating labels automatically⁶ during deployment and fine-tuning the detector, in case a user wants to train and evaluate one of the networks proposed in [43].

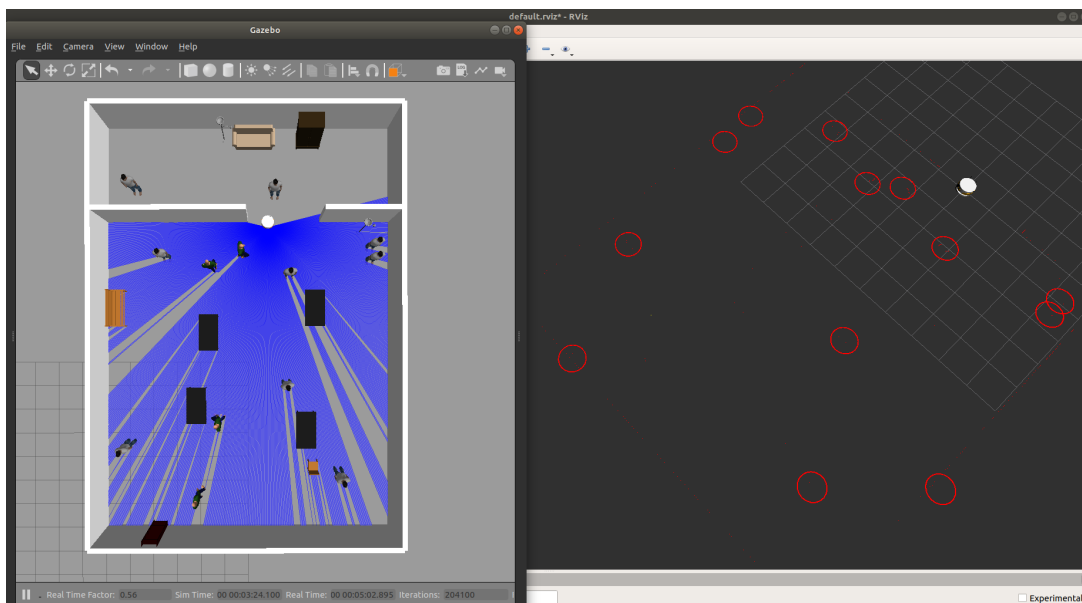


Figure 3.4: Simple scene of a home environment, inspired by the simple office with people context developed in [9]. The GAZEBO simulation on the left shows the context, while the RViz window on the right displays the laser scan data as red dots, with people detection predicted by the detector of DR-SPAAM [41]. Although most of the detection are correct, some errors occur.

⁵For further details: https://github.com/VisualComputingInstitute/2D_lidar_person_detection.

⁶For this purpose a calibrated camera is needed in the robot. As a calibrated camera it is intended a camera in which the transformation from the camera to the LiDAR is known.

3.1.4 Pedestrian Simulator (PedSim)

PedSim⁷ (Figure 3.5) is a ROS package for a 2D pedestrian simulator based on SFM [38]. It is a package useful for robot navigation experiments with simple or crowded scenes that are hard to acquire in practice. Its main features are:

- Individual walking using SFM for very large crowds in real time;
- Group walking using the extended SFM;
- Robot agent with teleoperation allowed for the user. Robot interactions with individual agents are properly handled through SFM;
- Social activities simulation;
- Sensors simulation (e.g., point clouds in robot frame for people and walls);
- Extensible Markup Language (XML) file-based scene design;
- Extensive visualization, since it is an environmental representation tool that is used on RViz;
- Option to connect with GAZEBO for physics reasoning.

These packages have been developed in part during the EU FP7 project Social situation-aware perception and action for cognitive robots (SPENCER)⁸.

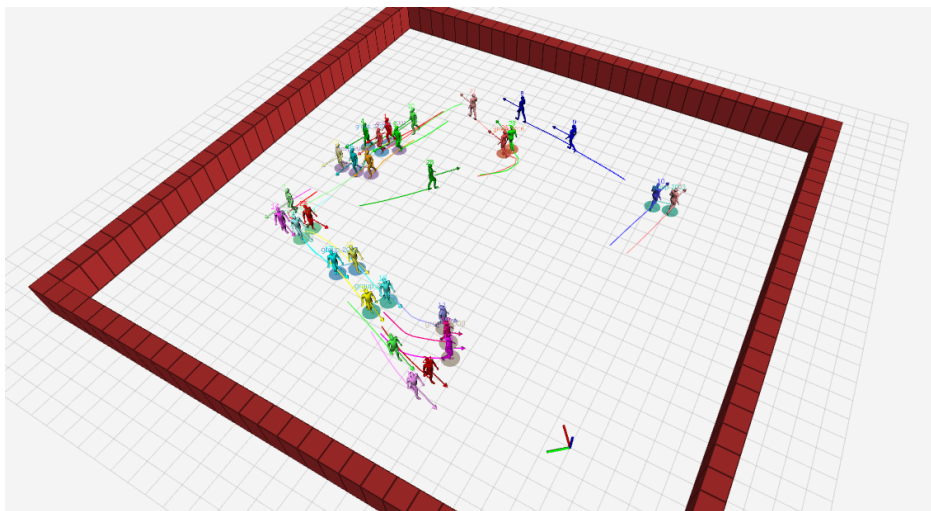


Figure 3.5: PedSim simulator simple scene.

⁷For further details: https://github.com/srl-freiburg/pedsim_ros .

⁸For further details: <http://www.spencer.eu/> .

3.1.5 OpenAI Reinforcement Learning (OpenAI RL)

OpenAI RL⁹ (Figure 3.6) is a ROS package that provides an RL set of libraries that allows users to train agents on user-defined tasks in needed environments.

Main components when working with the pipeline (Figure 3.6) of this package are:

1. *Training script*: it defines and sets up the learning algorithm that is going to be used to train the robot. It is composed by the learning algorithm that is provided in input to the `start_training` script, which runs the whole process;
2. *Task environment*: it specifies the specific task the robot has to learn;
3. *Robot environment*: it integrates the GAZEBO simulations of the robot and the algorithm environments;
4. *Gazebo environment*: this class connects the user's OpenAI programs to GAZEBO simulator.

These components can be grouped into two high-level classes: *Training script* (1) and *Training environments* (the components 2, 3, and 4 of the previous list).

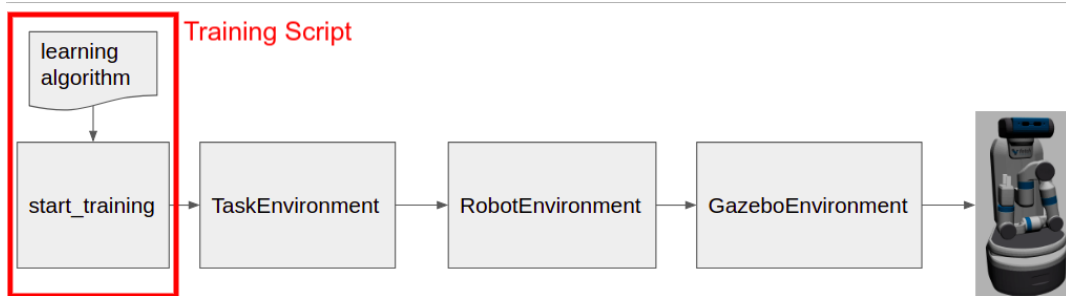


Figure 3.6: Pipeline followed by OpenAI RL for ROS.

⁹For further details: https://wiki.ros.org/openai_ros .

3.2 Take It And Go (TIAGo++) robot

TIAGo¹⁰ is a service robot designed to work in indoor environments. TIAGo's features make it the ideal platform to assist in research, especially in ambient assisted living or light industry. It combines mobility, perception, manipulation, and human-robot interaction capabilities.

TIAGo Mobile Manipulator Robot is produced by PAL Robotics which provides customizable modular configurations. The most popular are the following (Figure 3.7):

- TIAGo Base (a base robot without a body);
- TIAGo IRON (a robot with 1 screen to interact with the robot);
- TIAGo STEEL (a robot with 1 arm and 1 hand made up of 1 gripper);
- TIAGo TITANIUM (a robot with 1 arm and 1 hand made up of 5 fingers);
- TIAGo++ (a robot with 2 arms and 2 hands).

Modular Configurations



Figure 3.7: TIAGo Mobile Manipulator Robots Modular Configurations by PAL Robotics. From left to right: TIAGo Base, TIAGo IRON, TIAGo STEEL, TIAGo TITANIUM, TIAGo++.

3.2.1 Versatility

TIAGo is also very versatile since it can work in a lot of fields such as health-care, smart homes, collaborative robotics, Internet of Things (IoT), assisted living and research. A graphical representation of this property is in Figure 3.8.

¹⁰For further details: <https://wiki.ros.org/Robots/TIAGo> .

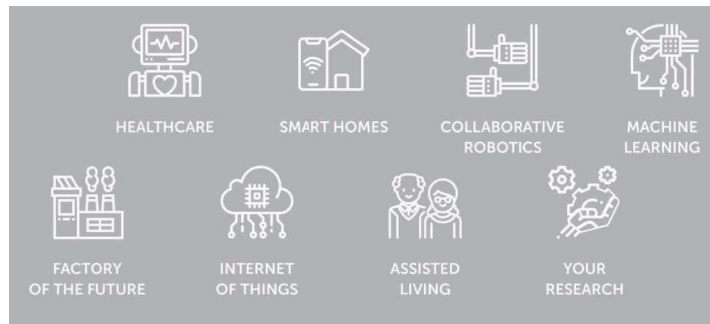


Figure 3.8: Fields in which Tiago can be applied.

3.2.2 Hardware architecture

TIAGo combines mobility, perception, manipulation, and human-robot interaction capabilities. The main parts of the robot used for real experiments are depicted in Figure 3.9. As explained in [67], its hardware architecture is modular and composed of:

- **Mobile base:** it has a differential drive mechanism. The base is provided with a laser plane for SLAM and safety purposes. On the rear part of the mobile base, there are some ultrasounds to prevent collisions when moving backward. TIAGo is capable of auto-navigating with a maximum speed of up to 1m/s;
- **Torso:** the upper body of TIAGo is composed of a lifting torso with a stroke of 35 cm so that the robot height can be adjusted between 110 and 145 cm. On the upper lateral part of the torso, there is a user panel providing some expansion ports. The top part of the torso (the so-called “laptop tray”) is flat so that a laptop can be placed on top of it. The frontal part has a stereo microphone and a speaker, both aimed at Human-Robot Interaction. The robot used for the experiments has two arms;
- **Arms:** the arms of TIAGo++ have 7 Degrees of Freedom (DoF). A force/-torque sensor can be attached at the end point of the wrist;
- **End-effectors:** in the TIAGo++ robot the common end-effectors are a parallel gripper and/or a robotic hand. The one proposed by PAL Robotics is the Hey5 hand;
- **Head:** it has 2 DoF providing 2 degrees of movement and it is equipped with a RGB-D camera.

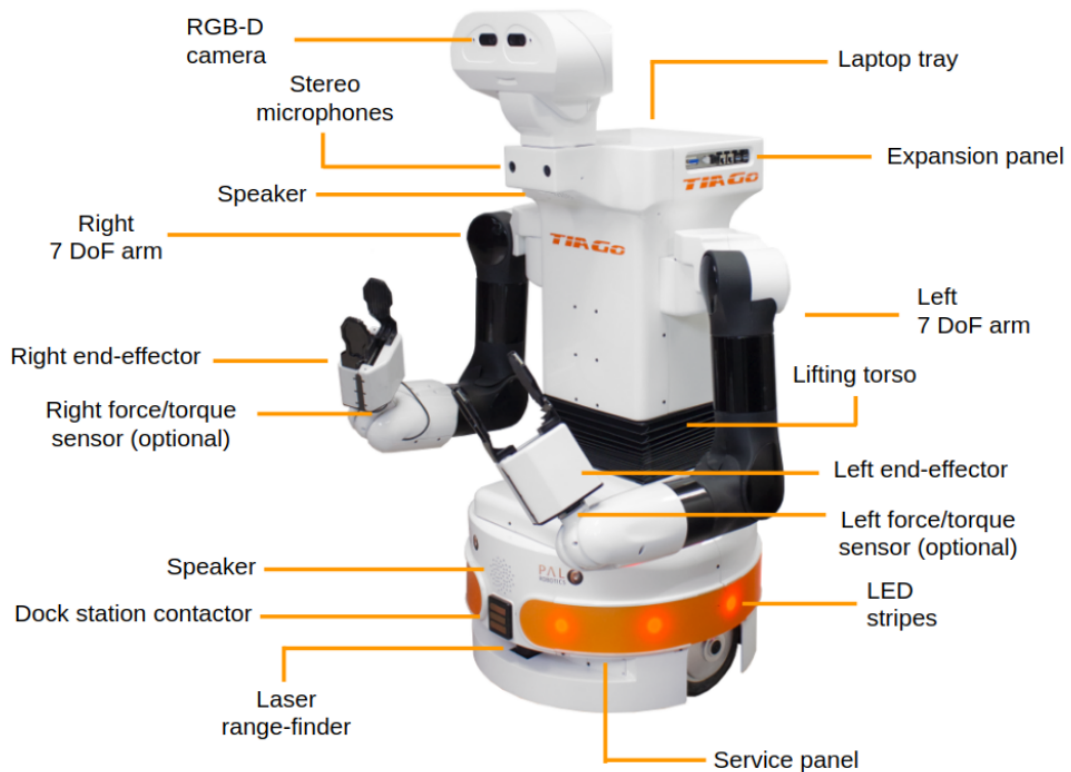


Figure 3.9: TIAGo++'s main components.

All these parts can be moved by different motors that can be controlled using ROS interfaces. For instance, TIAGo's head has 2 DoF, so we can move it up to down and left to right, and with this, we can perform an environment scan to map the scene. In addition, to perform manipulation tasks the lifting torso can be used to increase the workspace by positioning TIAGo in a higher or lower place or taking the arm to a vertical movement.

For navigation purposes, it is worth noting that this robot is equipped with several sensors such as LiDAR and RGB-D cameras. Specifications of the robot are summarized in Table 3.1.

3.2.3 Software architecture

The software architecture of TIAGo is modular. TIAGo uses different software middlewares to take the best from each one and provide a powerful software architecture. Built on top of the Ubuntu Operating System, the most interesting components for this thesis are the following:

Dimensions	Height	110 – 145 cm
	Weight	72 Kg
	Base footprint	Diameter 54 cm
Degrees of freedom	Mobile base	2
	Torso lift	1
	Arm	4
	Wrist	3
	Head	2
	Hey5 hand	19 (3 actuated)
	PAL gripper	2
	Drive system	Differential
Mobile base	Max speed	1 m/s
Torso	Lift stroke	35 cm
Arm	Payload	2 Kg
	Reach	87 cm
Electrical features	Battery	36 V, 20 Ah
Sensors	Base	Laser range-finder
		Sonars
		IMU
	Torso	Stereo microphones
	Arm	Motors current feedback
	Wrist	Force/Torque
	Head	RGB-D camera

Table 3.1: TIAGo++’s main specifications.

- ROS: the real standard robotics middleware that provides well-known interfaces to roboticists to develop applications;
- `ros_control`: a specific layer of ROS to provides access to the hardware exposing ROS interfaces. All the controllers of TIAGo are implemented as plugins of `ros_control` so that the users can develop their plugins and replace the default controllers;
- MoveIt!¹¹: the motion planning library comes off-the-shelf and integrated with TIAGo, so that users can easily develop applications requiring complex upper body motions, preventing self-collisions and collisions with the environment.

¹¹For further details: <https://moveit.ros.org/> .

Furthermore, TIAGo is provided with whole-body control. More precisely, its hierarchical quadratic solver provides online inverse kinematics of the robot's upper body:

- 7 DoF arm;
- Torso prismatic joint;
- 2 DoF head;
- Self collision avoidance;
- Joint limit avoidance;
- Gaze control.

Its code is used in running and simulating the robot with several GAZEBO-simulated environments. TIAGo's ROS package is an elaborate package defined for executing several robotic purposes like auto navigation, teleoperation, joystick control, and keyboard control. It has a camera package for image processing purposes. The package also contains arm control, head control, and torso control.

Finally, TIAGo works with open software and custom packages, URDF models, drivers, and controllers. All of them are available online.

3.2.4 Simulated TIAGo Base robot

Since the complete TIAGo model results to be a bit heavy when simulated, for this thesis the TIAGo Base Robot has been preferred to be used in all simulations. In this way, a lot of components like the body, the arm, and the head are not included so that they do not need to be simulated and handled, resulting in a much less heavy computation load. The appearance of the robot in the GAZEBO simulator (Section 3.1.1) is shown in Figure 3.10, while the one in RViz (Section 3.1.2) is in Figure 3.11.

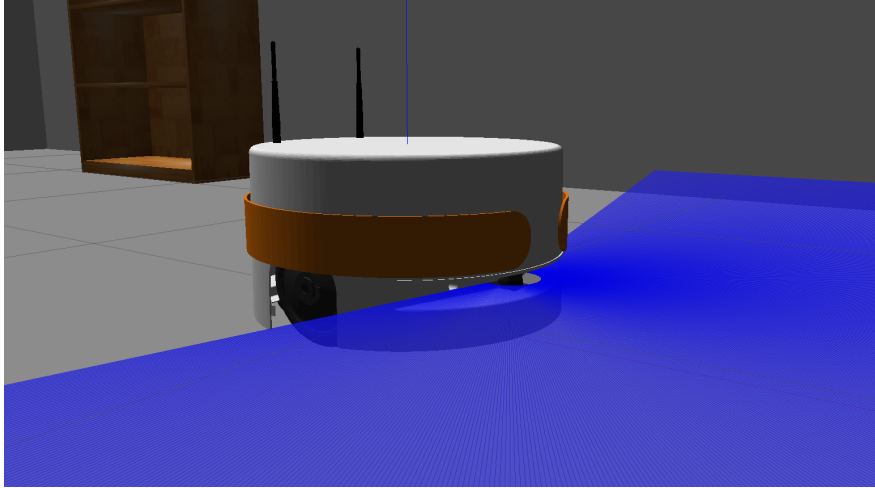


Figure 3.10: TIAGo Base Robot by PAL Robotics in GAZEBO simulator environment.

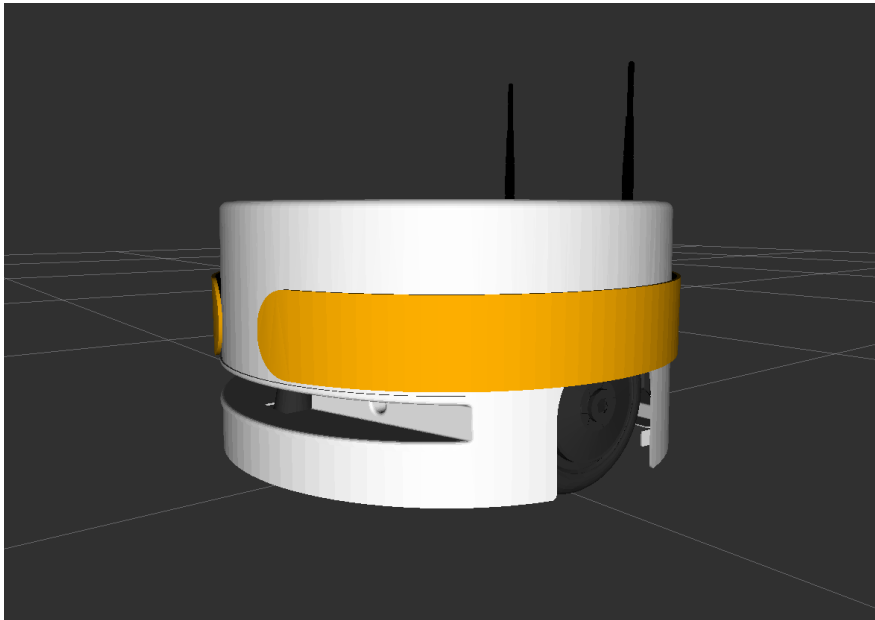


Figure 3.11: TIAGo Base Robot by PAL Robotics in RViz 3D environment.

Chapter 4

Preprocessing: People detection and laser points classification

In this section the people detection method is described, starting from the main geometrical technique to interpolate laser data with human feet, then the data points classification is described, to return the final people detection. The content of this chapter will be further used in Chapter 5 as preprocessing.

4.1 Obstacles interpolation method

The first phase in the people detection module used in this thesis is the interpolation of a detected obstacle using N laser data points $\{p_1, p_2, \dots, p_N\}$. Here, for simplicity, we assume that all N points belong to the same object¹. Given these points, the aim of this initial module is to interpolate them with a circle. The radius of each circle will be compared properly with a measure template of reference, computed in Section 4.2, and used as described in Section 4.3.

Given a set of points $P = \{p_1, p_2, \dots, p_N\}$ of an object detected by the laser, it is assumed to be a circular object when they represent human legs². Since with the laser only points on a circumference arc can be acquired, the computation of the object radius is required to get its center. As in Algorithm 5, given a set of points, the laser points of the object that are picked are $P_1 = [x_1, y_1]$, $P_C = [x_C, y_C]$, and $P_2 = [x_2, y_2]$. These are the points of first, middle and last

¹This can be assumed at this point since in the next steps it will be guaranteed by construction.

²This assumption is motivated because the laser detects objects and people at the human ankle height and the human ankle can be simplified as a circular shape.

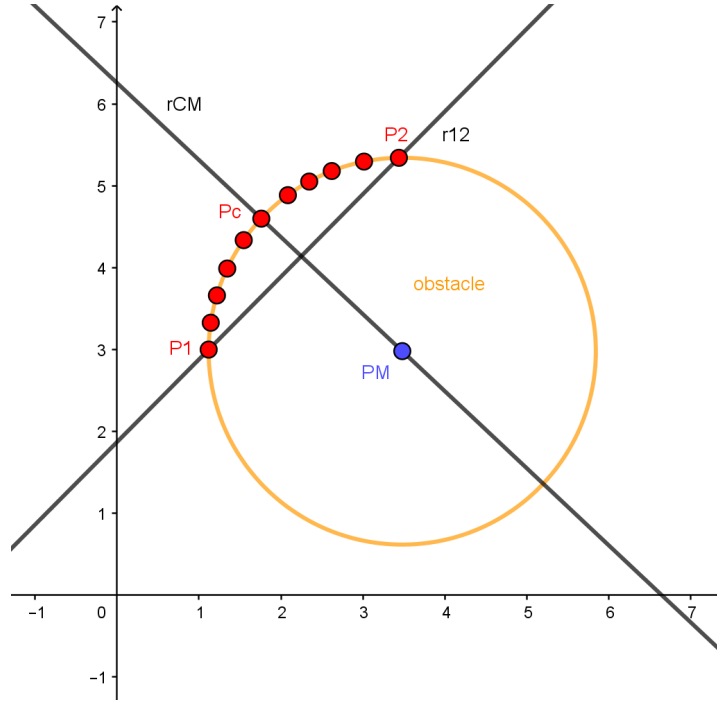


Figure 4.1: Graphical representation of the geometric interpolation from the laser data points related to a person's leg or an obstacle. Red dots represents laser data points. P_1 , P_C , and P_2 are respectively the laser points of first, middle, and last index. P_M is the center of the circle that will be approximated.

index. To compute the radius, the center point $P_M = [x_M, y_M]$ must be known. With basic geometry, imposing the equality on the radii $\overline{P_M P_1} = \overline{P_M P_C}$ that will be identified as Equation 4.1, knowing P_1 , P_2 , P_C , y_M (Equation 4.2), m_{CM} as the angular coefficient of the line $\overline{P_C P_M}$ (Equation 4.3) and developing the calculations, x_M can be obtained (Equation 4.4). In this way, the coordinates of the circle center P_M can be expressed as a function of only one variable x_M . A graphical representation of what is above is in Figure 4.1.

$$\overline{P_M P_1} = \overline{P_M P_C} \Leftrightarrow (x_M - x_1)^2 + (y_M - y_1)^2 = (x_M - x_C)^2 + (y_M - y_C)^2 \quad (4.1)$$

$$y_M = m_{CM} \cdot (x_M - x_C) + y_C \quad (4.2)$$

$$m_{CM} = -\frac{1}{m_{12}} = -\frac{x_2 - x_1}{y_2 - y_1} \quad (4.3)$$

$$x_M = \frac{(y_C - y_1)^2 + x_1^2 - x_C^2 - 2m_{CM} \cdot x_C \cdot (y_C - y_1)}{2(x_1 - x_C - m_{CM} \cdot (y_C - y_1))} \quad (4.4)$$

Algorithm 5 General circle interpolation *points_interpolation(P)*

Input: Points $P = \{p_1, p_2, \dots, p_N\}$ where each $p_i = P[i] = \{x_i, y_i, z_i\}$,1: $N = |P|$;**Output:** Center $P_M = (x_M, y_M)$, Radius $\overline{P_M P_1} = \overline{P_M P_C} = \overline{P_M P_2}$;

2:

Pick the main points.

3:

4: $P_1 = (x_1, y_1) \leftarrow P[1] = p_1$;5: $P_2 = (x_2, y_2) \leftarrow P[N] = p_N$;6: middle_index $\leftarrow \lfloor N/2 \rfloor$;7: $P_C = (x_C, y_C) \leftarrow P[\text{middle_index}]$;

8:

Compute the needed quantities.

9:

10: Impose $\overline{P_M P_1} = \overline{P_M P_C}$ (Equation 4.1);11: Compute y_M (Equation 4.2);12: Compute m_{CM} (Equation 4.3);13: Compute x_M (Equation 4.4);**return** $P_M = (x_M, y_M)$, $r = \overline{P_M P_1}$;

4.2 Template Computation for detecting people

This part handles the computation of the parameters for the measure template of reference mentioned in the previous section. The template used in this test represents a Gaussian distribution of radius values. More precisely, the template will be used to compare a given radius to the true radius that can be associated with a human foot. The idea behind is that a particular radius measure can be identified as a human foot radius with a certain probability. This probability increases when the radius value is close to the parameter mean μ_r of the template reference. To check how far a computed radius is from the reference mean, the parameter standard deviation σ_r of the template reference is introduced.

To get these two reference values, data acquisition has been performed using a simulated context in which there are only standing people that are placed at different distances and positions. The top and rear views of the scenario used are respectively in Figure 4.2 and Figure 4.3.

The procedure of data acquisition starts with the reading of laser data, then

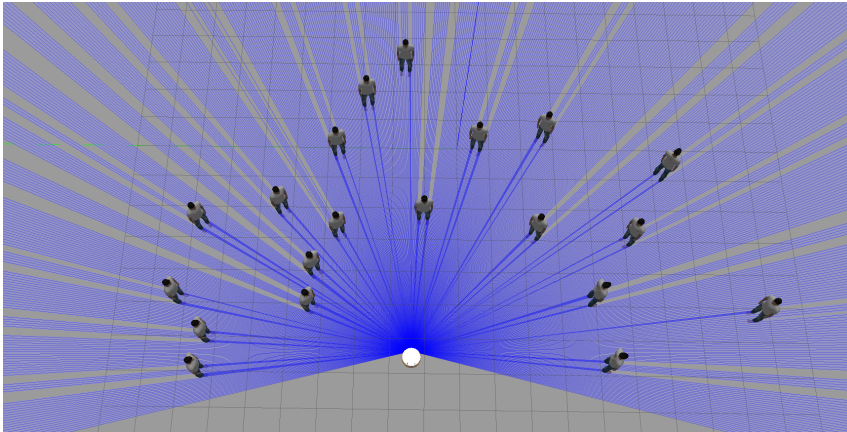


Figure 4.2: Top view of the crowded context in which some parameters of the gaussian based template used in this thesis are computed.

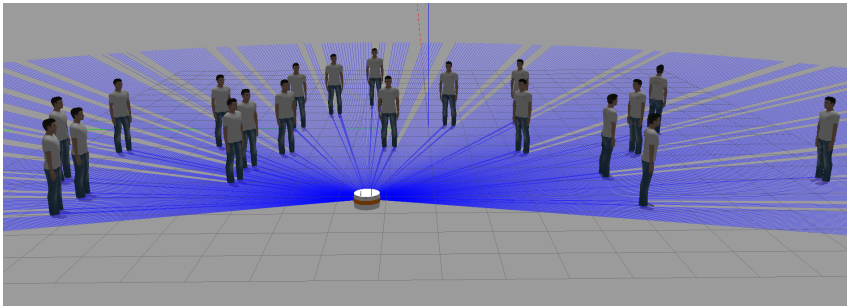


Figure 4.3: Rear view of the crowded context in which some parameters of the gaussian based template used in this thesis are computed.

it filters the not physical measures³ and converts them from polar to Cartesian coordinates. At this point, Algorithm 6 can be run.

In the first sub-task presented in Algorithm 7, the list of Cartesian points can be divided into groups of points, all belonging to the same object detection. The first threshold used is needed to group the points. It has been set enough bigger than the maximum distance between two consecutive laser points, hence with respect to the laser specifications (e.g.: angle of increment, maximum range, ...). The second important threshold defined regards the minimum number of points required, so that groups made up of only a few points are deleted, increasing the robustness of the procedure to outliers.

Then, the interpolation process in Algorithm 8 is applied to each group of points with a circle as defined in the Algorithm 5, otherwise in the case of only two points in a group, the radius is computed as half of their 2D distance. As soon as the radius of each group has been obtained, the mean and the standard

³If in a laser point $p(range, \alpha)$ the laser does not detect any obstacle in the distance range between its minimum and maximum range ($range_min \leq range \leq range_max$, α fixed), then the *inf* value is assigned.

deviation are returned and saved for further use (Section 4.2).

An example of the distribution obtained is in Figure 4.4. These parameters depend on the model of standing people used in the simulation.

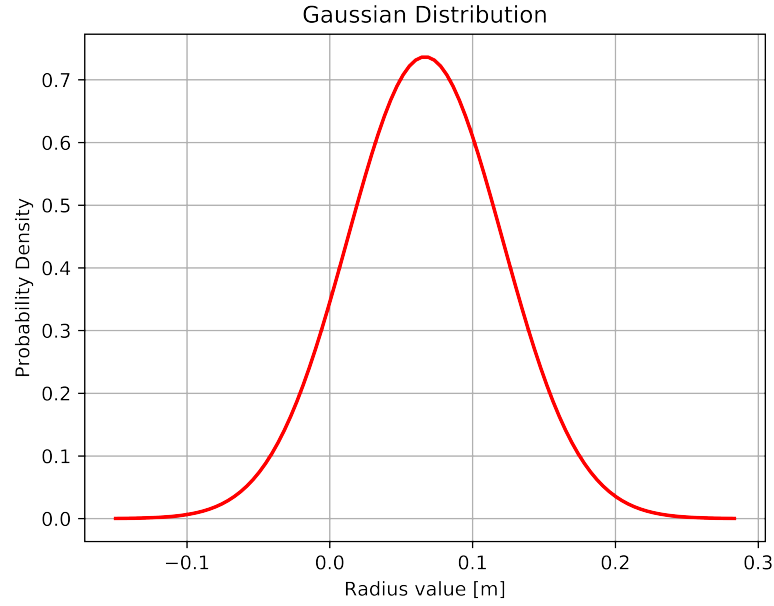


Figure 4.4: Example of Gaussian distribution obtained with the parameters (μ_r, σ_r) equal to $(0.066639, 0.054146)$.

Algorithm 6 Human Feet Detection Parameters

compute_parameters(P)

Input: Points $P = \{p_1, p_2, \dots, p_M\}$ where each $p_i = P[i] = \{x_i, y_i, z_i\}$,

$M = |P|$ (Each p_i is a laser point. People's positions may not be included in the list of points);

Output: Mean μ_r and standard deviation σ_r of a Gaussian distribution function;

CONST *MINIMUM_DISTANCE_OBJECTS* = 0.01 m;

CONST *MINIMUM_NUMBER_OF_POINTS* = 2;

obstacle_data_groups \leftarrow *group_data_points*(P);

$\mu_r, \sigma_r \leftarrow$ *compute_template_parameters*(*obstacle_data_groups*);

return mean μ_r , standard deviation σ_r ;

Algorithm 7 Group Laser Data Points*group_data_points(P)*

Input: Points $P = \{p_1, p_2, \dots, p_M\}$ where each $p_i = P[i] = \{x_i, y_i, z_i\}$,
 $M = |P|$ (Each p_i is a laser point. People's positions may not be included in the list of points);

Output: Groups of valid points $g = \{g_1, g_2, \dots, g_{F'}\}$ where each group
 $g_i = \{p_{1_i}, p_{2_i}, \dots, p_{N_i}\}$ with $1 \leq i \leq F'$. In general: $N_i = |g_i|$, $N_i \neq N_j$,
 $\forall i \neq j, 1 \leq i, j \leq F'$

Separate points into groups: each group contains points belonging to the detection of the same human foot.

```

for  $i \leftarrow 1$  to  $M - 1$  do
  save_point_in_current_foot(P[i]);
  if  $\text{distance}(P[i], P[i + 1]) > \text{MINIMUM\_DISTANCE\_OBJECTS}$  then
    save_current_foot();
  end if
end for
save_last_foot();

```

At this point the M points have been divided into F groups, where each group $g_i = \{p_{1_i}, p_{2_i}, \dots, p_{N_i}\}$ with $1 \leq i \leq F$.

In general: $N_i = |g_i|$, $N_i \neq N_j, \forall i \neq j, 1 \leq i, j \leq F$ (this highlights that it is not needed to have groups with the same number of points).

Delete groups made up of only a few points (less than 2).

```

for  $i \leftarrow 1$  to  $F$  do
  if  $N_i \geq \text{MINIMUM\_NUMBER\_OF\_POINTS}$  then
    save_current_foot();
  end if
end for

```

```

return Groups of valid points;

```

Algorithm 8 Interpolate Group Points

compute_template_parameters(g)

Input: Groups of valid points $g = \{g_1, g_2, \dots, g_{F'}\}$ where each group $g_i = \{p_{1_i}, p_{2_i}, \dots, p_{N_i}\}$ with $1 \leq i \leq F'$.In general: $N_i = |g_i|$, $N_i \neq N_j, \forall i \neq j, 1 \leq i, j \leq F'$ **Output:** Mean μ_r and standard deviation σ_r of a Gaussian distribution function;

The groups obtained at this stage are in number F' . Interpolate each group of points with a circle.

sum_radii \leftarrow 0;**for** $i \leftarrow 1$ to F' **do** **if** $N_i \geq 3$ **then** *It means that group g_i contains at least 3 points.* $r_i \leftarrow$ points_interpolation(g_i); *The points_interpolation(P) algorithm is used.* **else if** $N_i = 2$ **then** *It means that group g_i contains 2 points ($g_i = \{p_{1_i}, p_{2_i}\}$).* $r_i \leftarrow$ distance($g_i[1], g_i[2]$) / 2; **end if** save_current_radius(r_i); *The current radius has been saved in all_radii that has size R.* sum_radii \leftarrow sum_radii + r_i ;**end for***Compute the final parameters* $\mu_r \leftarrow$ sum_radii / F' ;deviation \leftarrow 0;**for all** radius \in all_radii **do** deviation \leftarrow deviation + (radius - μ_r)²;**end for** $\sigma_r \leftarrow \sqrt{\text{deviation}/R}$;**return** mean μ_r , standard deviation σ_r ;

4.3 People Detection method

At this stage, everything is ready to apply the algorithms above to detect people in given contexts. Four important measures have to be defined. The normalization factor for the Gaussian distribution (Equation 4.5) to make it a probability density function, and the two thresholds to decide regions of confidence for that Gaussian. In addition, the two defined in Algorithm 6 are considered. The laser data are read from a scan topic (input of Algorithm 13) and the parameters (μ_r, σ_r) of the Gaussian distribution computed in Section 4.2 are loaded. All laser measures are verified to be physical measures and converted from polar to Cartesian points.

$$f(x; \mu_r, \sigma_r) = NORM_FACTOR \cdot \frac{1}{\sigma_r \sqrt{2\pi}} e^{-\frac{(x-\mu_r)^2}{2\sigma_r^2}} \quad (4.5)$$

After, all points are separated into groups of points, each one representing a detected obstacle. As mentioned in Algorithm 6, the robustness is increased by deleting groups composed of only a few points, with the difference that here two new laser data messages are initialized, according to the level of confidence associated to them. Since one of the goals is to publish messages of laser points to evaluate and return at the end of Algorithm 13, one message will be used to store laser data belonging to high-confidence people detection, while the other will contain laser data of middle-confidence people detection. The information in the second type of message will be used as a starting point to guide the active behavior of the robot (please refer to Chapter 5). The messages will be initialized by copying the original laser message, with only ∞ data since they are not physical data, to be overwritten with real data in the next steps. Next, each group is interpolated with a circle for the same reason explained in the two sections above (Section 4.1, Section 4.2), in the same way as in Algorithm 6. In case the radius obtained is *NaN*, it is assigned the ∞ value.

Classification of the obstacle is now done by looking at which region the current radius belongs (i.e., in which zone between the thresholds it is included). To do so, the Gaussian distribution is divided into five regions as represented in Figure 4.5. If the radius belongs to the green region, it means the current group of points detected has high confidence to be the detection of a human foot, hence its points are converted into polar coordinates and saved in the appropriate message. If the radius belongs to the light yellow regions, the group points are saved in the message of points of middle confidence, to be analyzed further. As

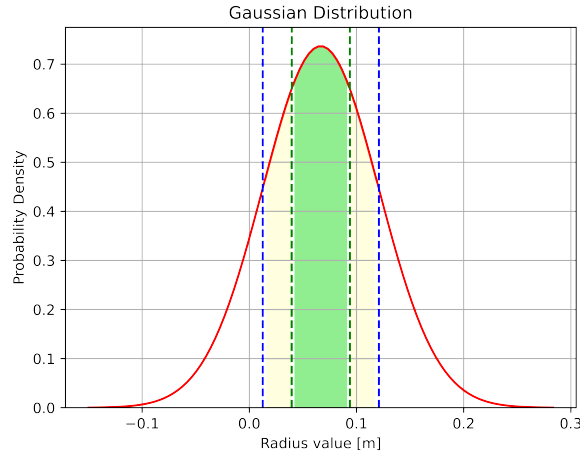


Figure 4.5: Example of classes of radii assigned to the classes for people detection: the green space refers to high confidence radii, yellow spaces refer to those of middle confidence, and not colored spaces to low confidence radii. In blue and green are displayed the thresholds to divide spaces. This Gaussian distribution is obtained with $(\mu_r, \sigma_r) = (0.066639, 0.054146)$. In this case thresholds used are $[\mu_r - 1.0 * \sigma_r, \mu_r - 0.5 * \sigma_r, \mu_r + 0.5 * \sigma_r, \mu_r + 1.0 * \sigma_r]$.

soon as all groups have been considered, the messages are published. This step ends with the classification part. The DR-SPAAM ROS detector (see Section 3.1.3) handles the detection by reading the laser messages that save the high confidence points (i.e., the points of the obstacles that have a high probability of being human feet detection) and by coupling the feet into people detection. In the end, also people's poses are available in the proper topic.

Algorithm 13 uses other smaller algorithms that specify each sub-task as in Algorithms 9, 7, 10, 11, 12.

Algorithm 9 Laser data to Points

laser_data_to_points(scan_topic)

Input: Laser scan topic name *scan_topic*;

Output: Array of physical points *P* in Cartesian coordinates;

```

M ← read_scan_topic(scan_topic);
for all range ∈ M.ranges do
  if range is a physical distance then
    P ← convert_polar_in_cartesian(M, range);
  end if
end for

return P;

```

Algorithm 10 Initialize new message

init_new_message(M)

Input: Laser scan message M ;

Output: Laser scan message *new_message* that is a copy of M , except for all ranges that are equal to ∞ ;

Initialize a new message to publish laser scan points.

$\text{new_msg} \leftarrow M$;

for all $\text{range} \in \text{new_msg.ranges}$ **do**

$\text{range} \leftarrow \infty$;

end for

return new_msg ;

Algorithm 11 Update Message to Publish

update_message(msg, g_i)

Input: Message msg with ranges data to update and the current vector of points g_i that belong to the same object detection;

Output: Message msg with ranges data updated;

for all $\text{point} \in g_i$ **do** *For each point in the current group g_i .*

$\text{range}, \text{angle} \leftarrow \text{convert_cartesian_in_polar}(\text{point})$;

$\text{index} \leftarrow \frac{\text{angle} - \text{msg.angle_min}}{\text{msg.angle_increment}}$;

$\text{msg.ranges}[\text{index}] \leftarrow \text{range}$;

end for

return msg with ranges data updated;

Algorithm 12 Classify groups*classify_groups*(g , new_scan_msg , sab_scan_msg)

Input: Array with groups of points g , messages to classify new_scan_msg and sab_scan_msg ;**Output:** The two messages new_scan_msg and sab_scan_msg classified;sum_radii \leftarrow 0;**for** $i \leftarrow 1$ to F' **do** **if** $N_i \geq 3$ **then** *It means that group g_i contains at least 3 points.* $r_i \leftarrow$ points_interpolation(g_i); *The points_interpolation(P) algorithm is used.* **else if** $N_i = 2$ **then** *It means that group g_i contains 2 points ($g_i = \{p_{1_i}, p_{2_i}\}$).* $r_i \leftarrow$ distance($g_i[0], g_i[1]$) / 2; **end if** **if** radius = NaN **then** radius \leftarrow ∞ ; **end if** *Here assign the class and check to publish messages*radius_class \leftarrow assign_class(radius, μ_r , σ_r); **if** radius_class = "LIKELY" **then** update_message(new_scan_msg , g_i); **else if** radius_class = "UNDEFINED" **then** update_message(sab_scan_msg , g_i); **end if****end for** **return** new_scan_msg , sab_scan_msg ;

Algorithm 13 People Detection and Classification

people_detection_classification(*scan_topic*, μ_r , σ_r)

Input: Laser scan topic name *scan_topic*, parameters μ_r and σ_r ;

Output: Laser scan messages *new_scan_msg* of points labeled as LIKELY, people detected with high confidence (confidence $\geq 35\%$) *people_detection_poses*, laser scan messages *sab_scan_msg* of points labeled as UNDEFINED;

```

CONST MINIMUM_DISTANCE_OBJECTS = 0.01 m;
CONST MINIMUM_NUMBER_OF_POINTS = 2;
CONST NORM_FACTOR = 0.1;
CONST STDDEV_THRESHOLDS[2] = {0.5, 1.0};

```

```

P  $\leftarrow$  laser_data_to_points(scan_topic);
g  $\leftarrow$  group_data_points(P);

```

The groups obtained at this stage are in number F'.

Initialize a new message to publish laser scan points LIKELY.

```
new_scan_msg  $\leftarrow$  init_new_message(M);
```

Initialize a new message to publish laser scan points UNDEFINED.

```
sab_scan_msg  $\leftarrow$  init_new_message(M);
```

Interpolate each group of points with a circle. new_scan_msg, sab_scan_msg \leftarrow classify_groups(g, new_scan_msg, sab_scan_msg);

Publish the two messages.

```
publish(new_scan_msg);
publish(sab_scan_msg);
```

Use the detector DR_SPAAM_ROS (Section 3.1.3) to get the poses of people, detected with high confidence.

```
people_detection_poses  $\leftarrow$  DR_SPAAM_ROS(new_scan_msg);
```

```
return new_scan_msg, people_detection_poses, sab_scan_msg;
```


4.4 Intermediary test to verify the preprocessing approach

In this chapter two methods for people detection have been presented: the DR-SPAAM detector described in Section 3.1.3 and the human feet detection above, based on the template test, in addition to the detector. To verify and prove the second proposed method, some analysis is required. Here, the two techniques are compared in some contexts, in each of them the confusion matrix⁴, the precision, the recall, and the accuracy indexes are computed for both.

The terms Actual 0 and Actual 1 in the figures of confusion matrices are related respectively to a person that it is not really placed in a position, and to a person that is really placed in a position. The same holds for both Predicted 0 and Predicted 1, with the difference that they are related to the predictions of the approach.

In each context, the configurations are slightly different since humans simulate people walking. However, these comparisons have to highlight the difference on false positive detection predicted by the only use of DR-SPAAM detector with respect to our approach.

4.4.1 Crowded environment

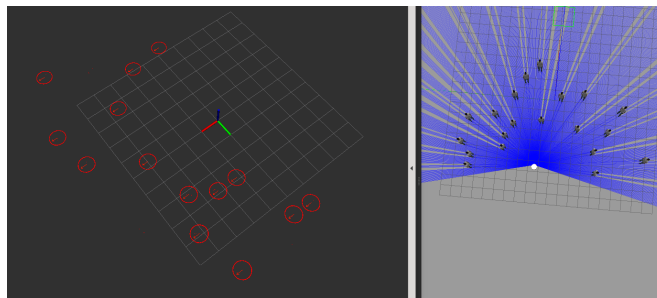


Figure 4.6: On the left: RViz window displaying the crowded environment with red circles indicating people detection is done only by the detector and the base_laser_link frame of the robot. On the right: GAZEBO window displaying the crowded environment with the robot.

The results of this comparison reward the simplest method (Figure 4.7). However, it must be highlighted that the context taken into account is highly controlled, since there are only people. For this reason, there is no chance to predict any obstacle as human feet, hence no false positive and true negative can occur.

⁴The values reported inside the cells of the tables are computed following the rule: given $Humans = \{h_1, h_2, \dots, h_H\}$, $cell = \sum_{i=1}^H h_i$ where $h_i \in \{0, 1\}$.

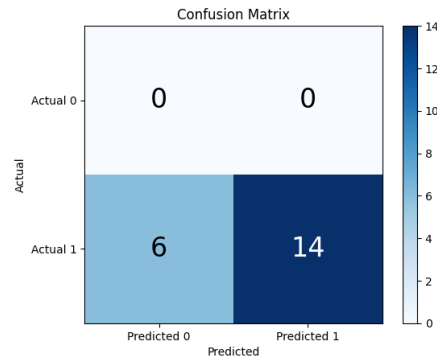


Figure 4.7: Confusion matrix of the crowded context where people detection is performed only by the DR-SPAAM detector. Actual values refer to the ground truth, while the others are to those predicted.

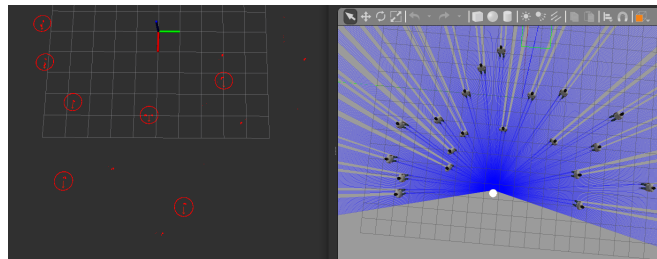


Figure 4.8: On the left: RViz window displaying the crowded environment with red circles indicating people detection is done by the human feet detector and the DR-SPAAM detector, and the base_laser_link frame of the robot. On the right: GAZEBO window displaying the crowded environment with the robot.

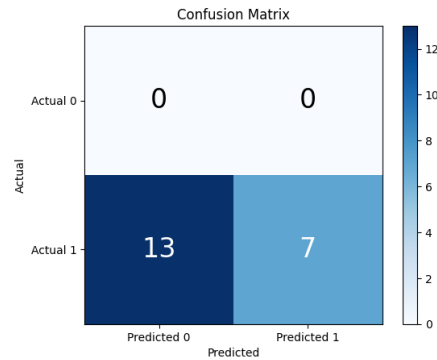


Figure 4.9: Confusion matrix of the crowded context where people detection is performed by the human feet detector and the DR-SPAAM detector. Actual values refer to the ground truth, while the others are to those predicted.

Indexes	DR-SPAAM only	Pre-processing + DR-SPAAM
Precision	1.00	1.00
Recall	0.70	0.35
Accuracy	0.70	0.35

Table 4.1: Table of indexes computed for both methods applied to the crowded environment.

4.4.2 No people environment

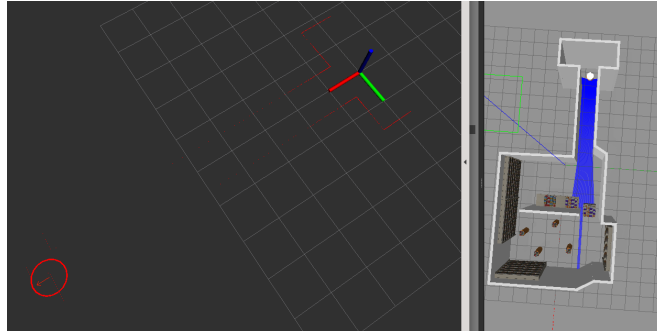


Figure 4.10: On the left: RViz window displaying the simple environment with no people, the red circles indicating people detection is done only by the detector and the base_laser.link frame of the robot. On the right: GAZEBO window displaying the simple environment with no people and the robot.

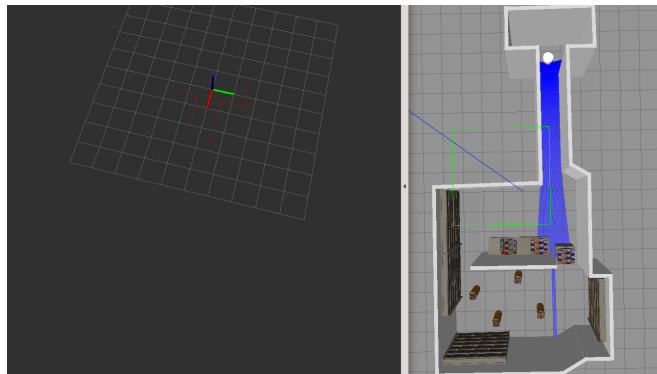


Figure 4.11: On the left: RViz window displaying the simple environment with no people and the base_laser.link frame of the robot. Here no people are detected by the human feet detector and the DR-SPAAM detector. On the right: GAZEBO window displaying the simple environment with no people inside and the robot.

In this test, although there are no people, we verify the performances of the two approaches. The results of this comparison seems to reward the second method (Figure 4.11). Indeed, this comparison has not been reported to compute the tables but to support the point highlighted before that the DR-SPAAM tends to over-detect. However, due to the simplicity, no confusion matrices are reported.

4.4.3 Simple office with people environment

Because of the complexity of the possible configurations of people in this environment, here more than one situation is taken into account. Each confusion matrix is computed summing all those computed for the respective situations. The results of this comparison favor the second method (Figure 4.14). However, it must be highlighted that the performances can be optimized by choosing more

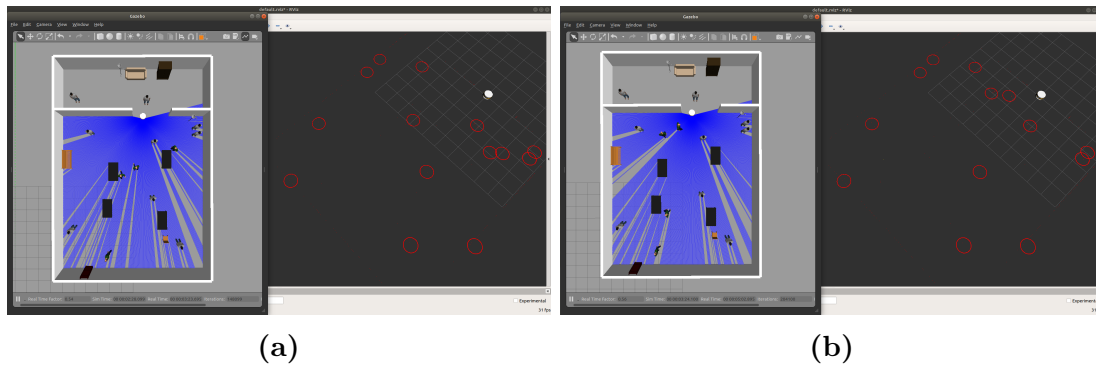


Figure 4.12: (a) and (b): configurations of people in the simple office environment. On the left of both pictures: GAZEBO window displaying the simple office environment with the robot, the people, and the furniture of the rooms. On the right of both pictures: RViz window displaying the simple office environment with red circles indicating people detection done by the DR-SPAAM detector and the base_laser.link frame of the robot. Note: only people in the FoV of the laser are considered for Figure 4.13 .

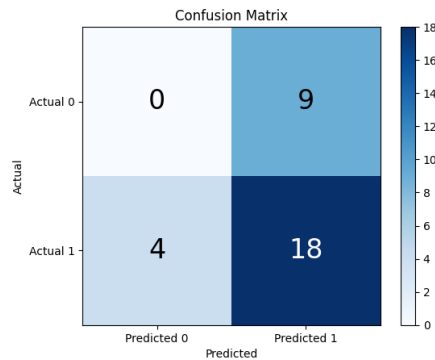


Figure 4.13: Confusion matrix of the simple office with people context where people detection is performed only by the DR-SPAAM detector. Actual values refer to the ground truth, while the others are to those predicted.

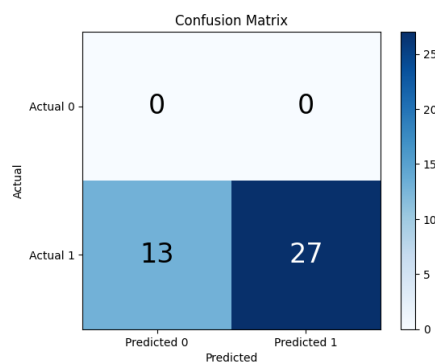


Figure 4.14: Confusion matrix of the simple office with people context where people detection is performed by the human feet detector and the DR-SPAAM detector. Actual values refer to the ground truth, while the others are to those predicted.

suitable parameters. In addition, more distant people are detected fewer times than those closer because of effects farthest introduced (i.e., occlusions, more noisy laser data), as expected. Finally, the zeros in the table in Figure 4.14 come

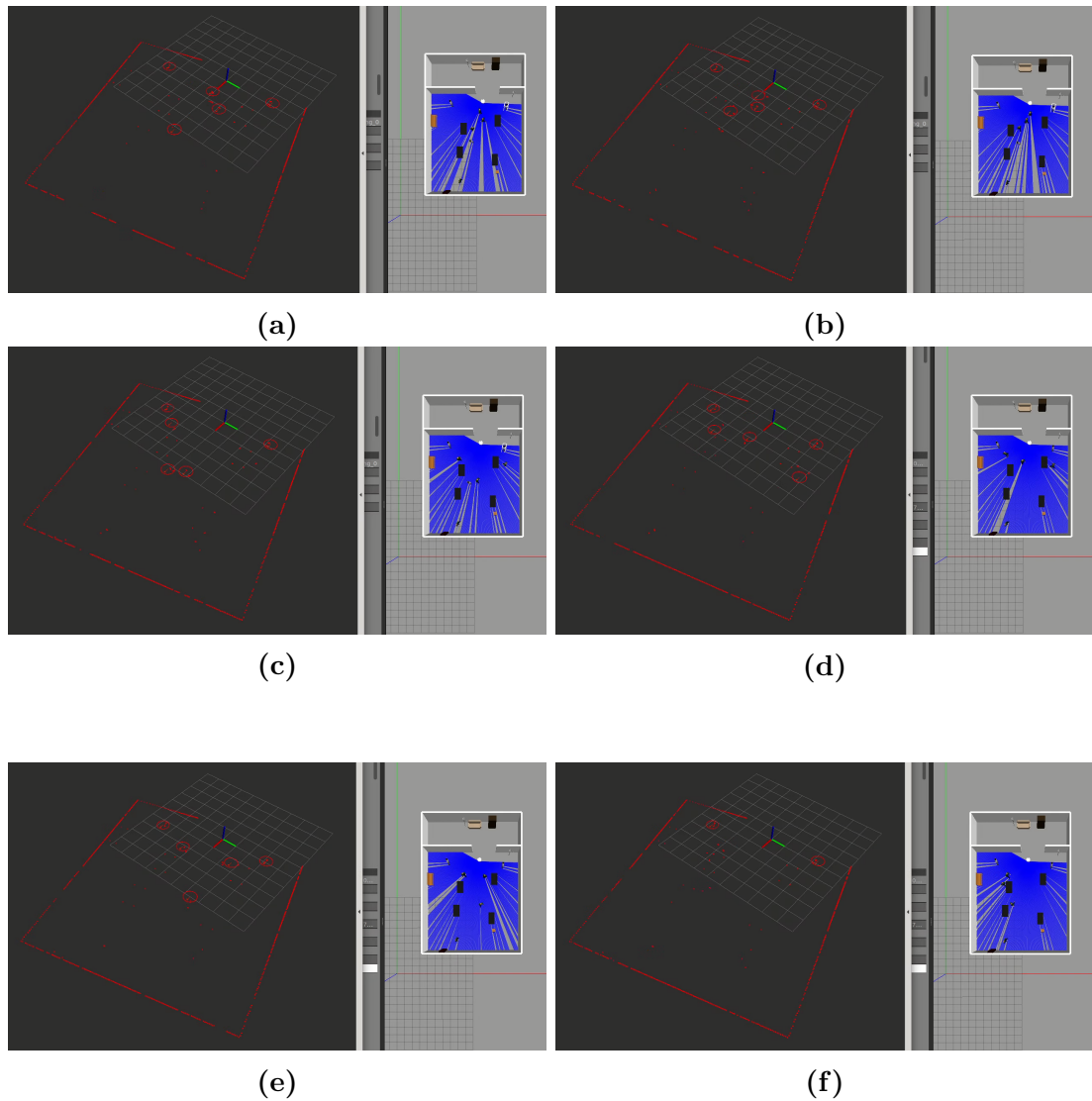


Figure 4.15: From (a) to (f): configurations of people in the simple office environment. On the left of each picture: RViz window displaying the simple office environment with red circles indicating people detection is done by the human feet detector and the DR-SPAAM detector, with the `base_laser_link` frame of the robot. On the right of each picture: GAZEBO window displaying the simple office environment with the robot, the people, and the furniture of the rooms. Note: only people in the FoV of the laser are considered for Figure 4.14 .

from the fact that the second detection method results in being more robust to outliers and over-predicts fewer times than the first one.

It is possible to notice that the combination of the pre-processing with the DR-SPAAM gives more importance to the people nearest to the robot. This is useful since people in the foreground occlude people behind them because of the laser operating principle. The fact that people closer to the robot can be detected

Indexes	DR-SPAAM only	Pre-processing + DR-SPAAM
Precision	0.67	1.00
Recall	0.82	0.68
Accuracy	0.58	0.68

Table 4.2: Table of indexes computed for both methods applied to the simple office with people environment.

more reliably is exploited in Section 4.5.

Given the observations reported above, the hybrid approach composed of preprocessing + DR-SPAAM has been chosen to be used in the following of this thesis work.

4.5 Discussion of the intermediary test

The performance of the detection method, the subject of this chapter, can be improved. The most direct ways to achieve this optimization are the following:

- In GAZEBO simulations, human models can be customized to be compliant with parameters computed in Section 4.2 (Figure 4.16). Hence, detection will be more accurate and precise since computations will respect the benchmarks (Figure 4.17);

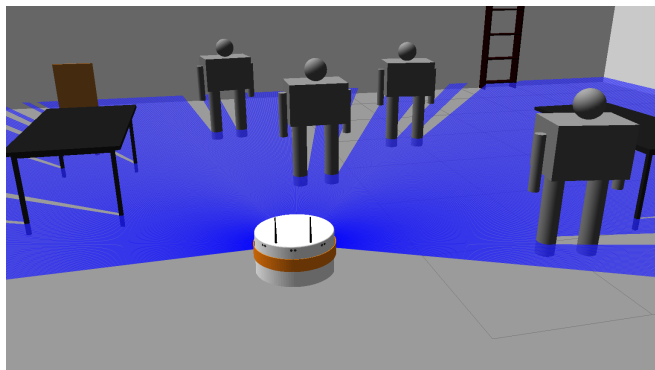


Figure 4.16: Custom human model for GAZEBO simulations. The most important links of this model are the legs, which respect the parameters (μ_r, σ_r) found in Section 4.2.

- Since parameters are computed through equations (Section 4.1), the more points, the lower approximation is obtained in the final parameters. The number of points changes due to the physical working principle of 2D LiDAR: the farther the sensor is from the object to detect, the more the laser will detect the object with fewer points. Hence, what changes with

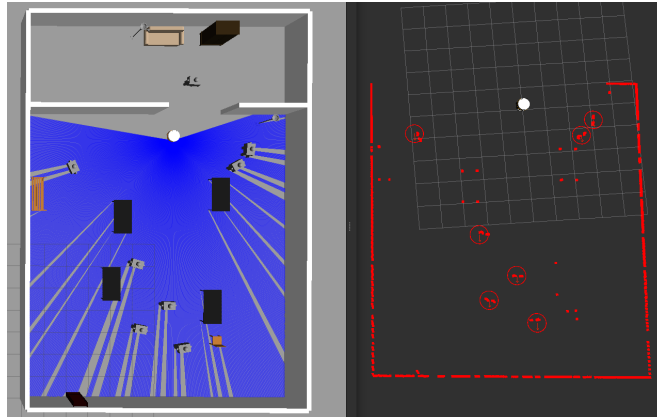


Figure 4.17: Custom human models that represent people detected in the simple office scenario. In this example detection predicted by the human feet detector and the DR-SPAAM detector are perfect. On the left, there is the GAZEBO simulation, while on the right the RViz window is displayed.

distance is the density of laser points that are reflected by the object (Figure 4.18, 4.19). This motivates the introduction of the starting principle of the Active Perception behavior developed in Chapter 5;

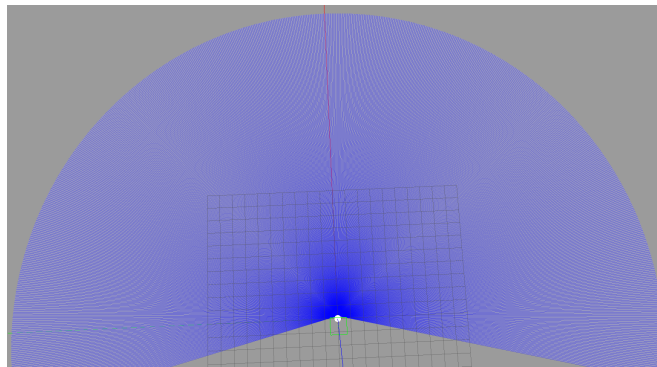
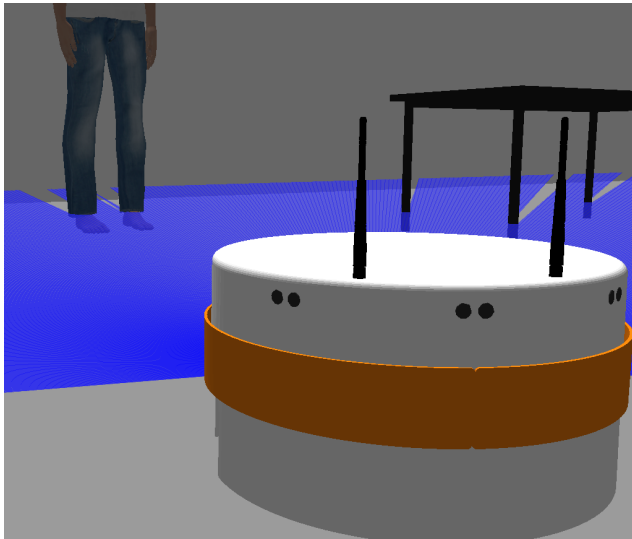
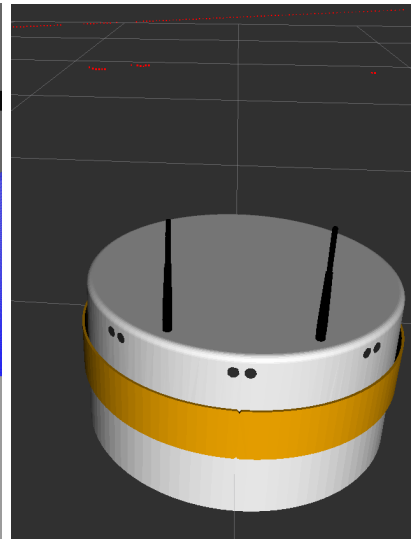


Figure 4.18: Simple GAZEBO simulation with 2D LiDAR: the density of laser rays gets smaller with distance from the robot.

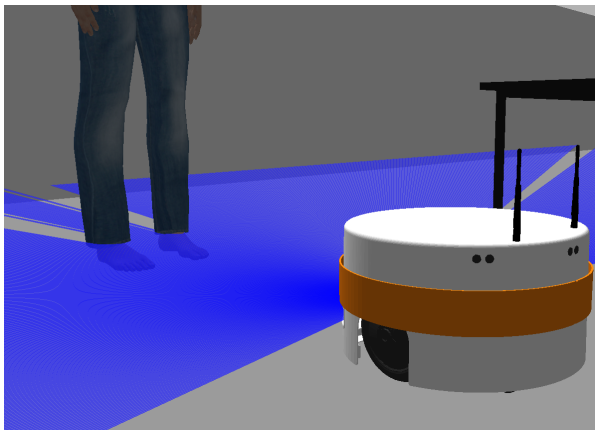
- In real applications this problem needs more reliable parameters. The most simple solutions can be datasets. Indeed, starting from a dataset that contains laser readings of people in crowded environments, the parameters can be estimated.



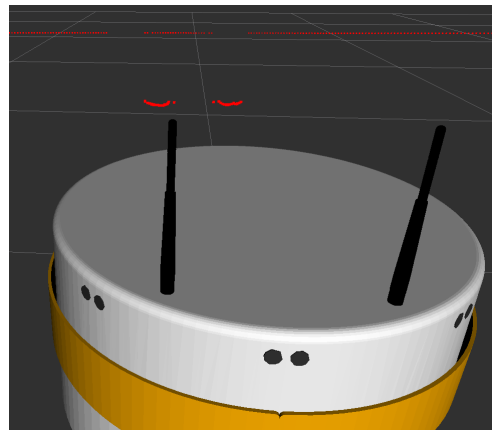
(a) The robot is far from the person (GAZEBO).



(b) The robot is far from the person (RViz).



(c) The robot is closer to the person (GAZEBO).



(d) The robot is closer to the person (RViz).

Figure 4.19: (a) and (b): the robot is far from the human and laser data are few. As soon as the robot gets closer to the person, distance decreases so laser data density increases. Indeed, in (c) and (d), there are many more points in the RViz window.

Chapter 5

Social Active Perception

In this current section, the methods developed for the robot to execute Social Active Perception behavior are reported in detail, starting from an overview on RL to the mathematical description of the deterministic active perception policy. For additional analysis of the topic, learning of this policy is proposed, followed by the RL training method developed for the agent.

The brief review about active perception reported in this chapter has been inspired by the work in [10].

The essence of active perception is to set up a goal based on some current belief about the world and to execute the actions that may achieve it. With the term perception, it is intended the environment in which an agent exists and how the agent interacts with that environment. This is strictly related to another term, that is affordance, which refers to the opportunities for action provided by a particular object or environment.

In the general concept, it is not needed to spend much effort and time on processing and artificially improving imperfect data, but rather accepting imperfect and noisy data as a matter of fact, and incorporating them into the overall processing strategy.

A general agent is an active perceiver if it knows why it wishes to sense, and then chooses what to perceive, and determines how, when, and where to achieve that perception [10]. Therefore, an actively perceiving agent (in this case the robot) is one that dynamically determines the why of its behavior and then controls at least one of the what, how, where, and when for each behavior. This is the reason why the five main constituents of an actively perceiving agent are defined: why, what, how, when, and where (Figure 5.1).

Each element of the active perception definition can be further decomposed

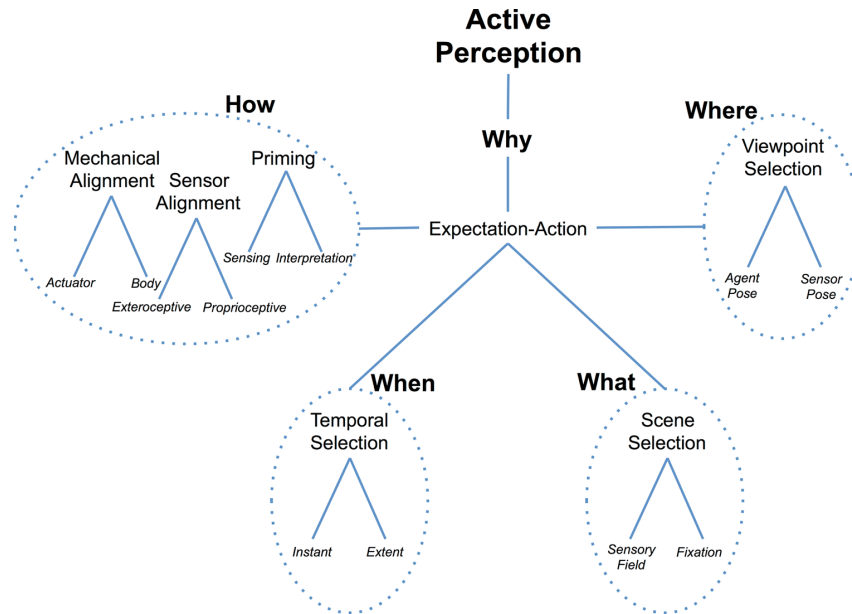


Figure 5.1: The basic elements of Active Perception broken down into their constituent components. Instances of an embodiment of active perception would include the Why component and at least one of the remaining elements whereas a complete active agent would include at least one component from each [10].

into the set of computations and actions it comprises, as shown in Fig. 5.1, noting that this decomposition is abstract and must be further detailed depending on each particular task.

In addition, resource constraints play an important role not only because of computer power and memory capacity but also because, in practice, the number of sensors is limited as well. Thus, choices must be made. The agent must be placed appropriately within the sensory field or, in other words, be mechanically aligned to its task. The sensing geometry must be set to enable the best sensing action for the agent's expectations. An agent that knows its body position and orientation concerning some arbitrary location can control the viewing angle and distance for each sensing act to place its sensors to best perceive the target and its aspects more relevant to accomplish its task.

A clear example of active perception performed by an agent in a mostly unknown environment covers both research operations in unknown places and the field of exploration for information gathering in environments not suitable for humans (e.g.: scientific missions to other planets [7]).

5.1 Reinforcement Learning (RL): Introduction

The main components of RL are reported in Figure 5.2. The basic process is:

- Observe of the environment;
- Decide how to act using some strategy;
- Act accordingly;
- Receive a reward or penalty;
- Learn from the experiences;
- Iterate over all episodes.

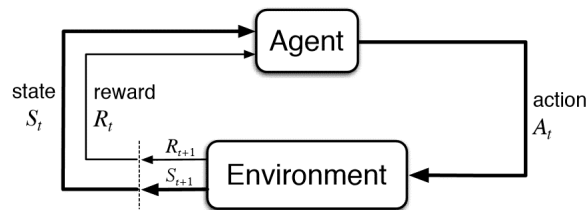


Figure 5.2: RL general scheme.

The strategy is also known as Policy π . The agent has to learn to act using the chosen policy, that is to decide on the action a to take, based on the current state s , with some probability (Equation 5.1).

$$\pi : (s, a) \mapsto [0, 1] : \pi(a|s) = \mathbb{P}(A = a|S = s) \quad (5.1)$$

The so-called state transition (Equation 5.2) is the transition that action a leads from the current state s_i to the next state s_{i+1} . In general, the randomness of the state transition is from the environment, while randomness in action is from the policy function (Equation 5.3) and the randomness in the state is from the state transition function (Equation 5.4).

$$p(s'|s, a) = \mathbb{P}(S' = s'|S = s, A = a) \quad (5.2)$$



$$A \sim \pi(\cdot|s) \quad (5.3)$$

$$S' \sim p(\cdot|s, a) \quad (5.4)$$

Given policy π , the action-value function $Q_\pi(s, a)$ evaluates how good it is for the agent to pick action a while being in state s (Equation 5.5).

$$Q_\pi(s, a) = \mathbb{E}[U_t | S_t = s, A_t = a] \quad (5.5)$$

The final basic process of RL training is precisely formulated in Algorithm 14.

Algorithm 14 General Reinforcement learning training

Input: Number of epochs n_epochs , number of steps n_steps

Output: Trained agent

```

1: for  $e \in n\_epochs$  do
2:   for  $t \in n\_steps$  do
3:     Observe state  $s_t$ ;
4:     Select action  $a_t \sim \pi(\cdot|s_t)$ ;
5:     Execute  $a_t$ ;
6:     Get new state  $s_{t+1}$  from environment;
7:     Get reward  $r_t$ ;
8:     Update the cumulative reward  $U_t$ ;
9:   end for
10: end for
    return Trained agent;
  
```

The rewards are used to guide the learning of the policy. Indeed, the final goal of RL is to maximize the expected future cumulative reward to ensure the agent learns as well as possible. Hence, Equation 5.6 is related to the cumulative reward (Equation 5.7).

$$U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots + \gamma^{n-t} R_n = \sum_{t=0}^n \gamma^t R_t \quad (5.6)$$

Here $\gamma \in [0, 1)$ is the discount rate: it is less than 1 to weigh the future events less than those in the immediate future. Instead, the current cumulative reward at step t can be simply computed as in Equation 5.7.

$$R_T = \sum_{t=0}^T R_t \quad (5.7)$$

5.2 Q-learning

The first idea was to use Q-learning since it is an important concept in the field of RL for the following reasons:

- Q-learning can be combined with deep neural networks to create DQN (Deep Q-Network). It is a deep neural network that is suitable for learning the quality function associated with a couple state-action in RL;
- It is a model-free RL approach, meaning it does not require a model of the environment;
- Q-learning can handle large state spaces effectively since it can use a table (Q-table) to store the values of the expected cumulative rewards for each state-action pair. Despite its simplicity, Q-learning can scale reasonably well to problems with a large number of states and actions.

During RL training, a Neural Network (NN) model learns how to act by associating the current state to the action that maximizes the reward. This association is captured through the update of the Q function, which stands for Quality function, and the parameters learning rate and discount rate. Figure 5.3 shows the mathematical equation to update the Q value for the pairs state-action: to the current Q values the reward for taking an action in a state is added by weighing it with the learning rate. Then, the discount rate weighs the difference between the maximum expected future reward that all possible actions could produce, starting from the Q values of the current state. Q-learning requires a finite number of actions¹ and states, but there are not finite states². Since Q-learning can not be always applied, more than one strategy to choose the proper actions must be found.

$$\text{New } Q(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \max_{a'} Q'(s', a') - Q(s, a)]$$

Figure 5.3: Q-learning equation.

¹The number of actions defined further will be finite.

²The states will be further defined to be a union of more data.

5.3 Introduction to the proposed approaches

In the beginning, the objective was to make the robot learn the abilities needed through RL training. This method would have needed the definition of elements like the state, the actions, the model of the network to train, some optimizers, a lot of hyperparameters, and a reward function. This raised the need to face some difficulties, which had to be handled mainly for the reward function definition. This function would have kept into account state data. Indeed, after some trials, the next step has been to define a custom reward function for RL training with finite motion actions. This process outputs a model able to map the state to the action to execute.

For comparison, the other part of the work has produced a mathematical formulation of a deterministic policy that the agent must follow and execute to act accordingly and reach its goal. The final general goal is always to maximize the number of people detected.

To make the active behavior to be compliant with the policy, the work started before collecting the intuition behind the approach, and then the implementation with the mathematical formulation of the policy that is driven by utility, reward function, etc. These elements are further presented in the next section, while the RL training guided by the reward function is explained in detail later.

5.3.1 Intuition behind the proposed approach

The robot starts a training episode. It should acquire people detection for some time interval³ and, if needed, it should create a distribution map (or something similar). The robot has to decide which action to take by computing the utility function for all possible actions, and then by choosing the one with max utility. After the chosen action has been performed, the robot repeats again the detection, and the reward function computation to get its current reward for the update of the total reward. Then, the loop goes on with the next iteration with the next action.

This happens for a certain number of episodes where each one lasts up to the maximum number of steps.

³This is not necessary, but it can increase reliability since a new person may be detected but only once, because of some reason like to be a false positive.

Here the main situations to take into account are the following:

- From the state the robot can compute the distribution of people;
- The robot should be penalized for taking too much time on an episode;
- The robot should obtain a higher reward if the variation in people detection is increasing;
- The robot should obtain a penalty (negative reward) as soon as it acquires fewer people than in the previous steps since if time passes and the robot loses detection, which results to be poor-quality work;
- The robot should choose the action that maximizes the trade-off between going towards a high-density cluster of people that is far away and going towards a non-maximum (a general number) of people in the cluster that is the closest.

5.4 Design and Implementation of the Proposed Approaches

In this section, a summary of the key contributions of this work is detailed. In the following figures, each block represents an important module for the whole pipeline. The color of the block specifies whether it represents the implementation of a work already available (grey) or new since some modifications were required in the existing modules (light red). All these pipelines have been aimed at defining the behavior of the robot, which is always the endpoint. All bulleted lists gather the blocks' main functionalities, but they are not always the only ones that a block can have.

5.4.1 Deterministic Policy

The framework for active perception guided by the policy in Figure 5.4 can be resumed into three main parts:

- *People detection and laser points classification*: the people detector implements the open-source DR-SPAAM for ROS, based on NN prediction. The classification task is based on the optimized parameters for crowded

contexts. This big block interpolates laser points to detect possible obstacles and it returns the input for the active perception task after obstacle filtering;

- *Navigation*: the planner of the ROS navigation stack with the use of a custom action server to execute the planned path from the current pose to the goal one;
- *Active Perception*: first, policy works on input data, indeed clustering and data memory are needed to associate an utility score to each possible action, so that the action that maximizes this score can be chosen. Finally, the execution of the action completes the task, from the state update to the evaluation of the action by the reward computation.

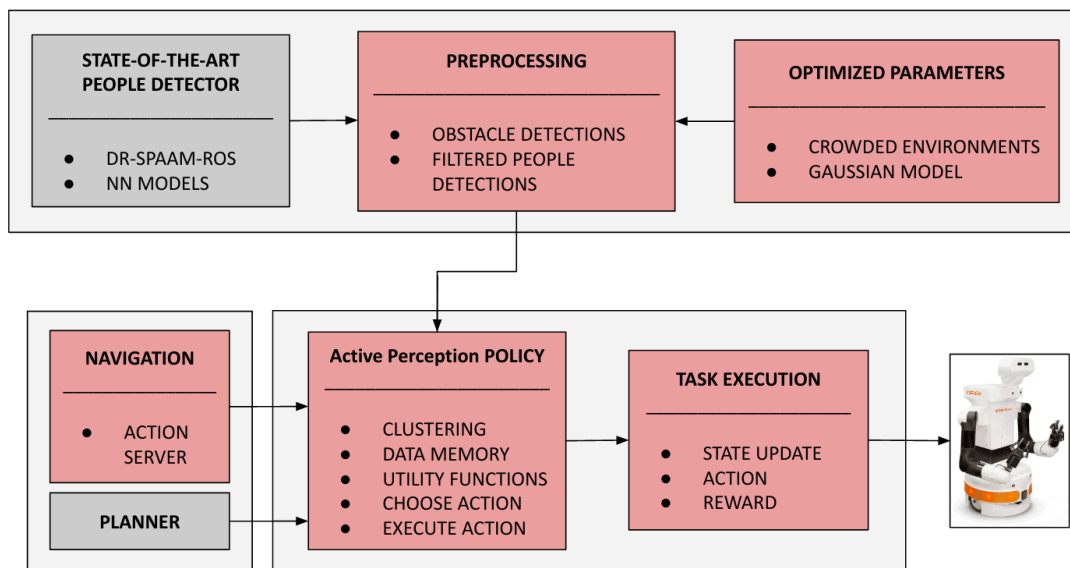


Figure 5.4: Scheme of the architecture for the active perception guided by the policy.

The interaction between perception and decision-making is the central aspect. Determinism is a property of the latter, while the former is not, since it is based on network predictions. As a consequence, given the same input, the policy returns the same action, but it is not guaranteed to get the same detection in the same situation.

Moreover, the pipeline is ready to use on the robot without any training. By looping it for a large number of runs, meaningful data can be acquired to be further learned by NN models. In this way, the focus is moved on how better networks can simulate this final process (Section 5.4.2).

5.4.2 Learning from data

Figure 5.5 shows the extension of the pipeline above. This time perception models can be obtained through data-driven approaches:

- *NN*: once the model has been defined with its forward learning method, a specific network operates;
- *Training*: it returns models able to simulate as best as possible the whole Active Perception block in Figure 5.4. Some metrics suggest how good has been the training phase.

Finally, trained models can work directly on the robot. As data-driven methods, results improve with the increase of data relating to more scenarios, more runs, and more contexts of use.

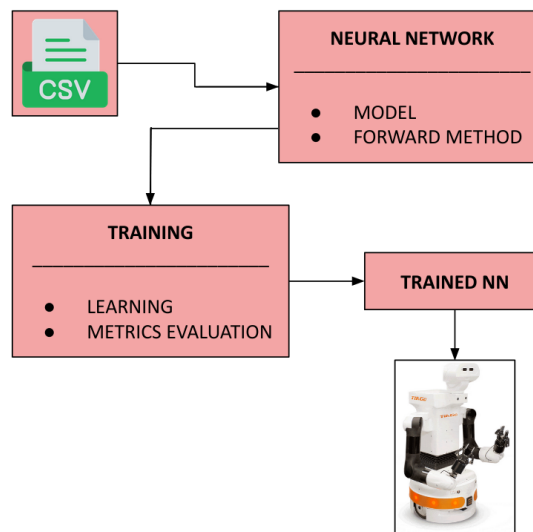


Figure 5.5: Scheme of the architecture to train active perception models to learn from data of policy executions.

5.4.3 Reinforcement Learning

The RL training can be split into training script and training environment. The pipeline in Figure 5.6 represents the application of advanced ML techniques in the field of active perception. As introduced in Section 3.1.5, the training environment is based on OpenALROS. It has been properly customized to adapt the base code to the current task. Instead, the training script defines the modules to handle the whole training:

- *Training script*: the network learns through the learning algorithm. Given the DQN structure and the forward method, the learning algorithm implements the Q-learning with batch data memory. The main functions such as Reset, Step, Done, and Close guide the training. In the end, the evaluation of the metrics returns the quality of the whole process;
- *Training environment*: this is the low-level handler of the whole pipeline. More precisely, in the Task environment the actions, the reward, the clustering, and the done are performed and checked. Their execution changes according to the robot used. For this reason, the next step is to send the commands to the Robot environment, where the set-up of the sensors and the collision check are defined. The final step is to execute the program. Since the training is simulated in Gazebo, a proper module called Gazebo environment connects everything. As soon as the high-level functions are called, the module associates data incoming from the robot's sensors with the definitions of the functions, so that the proper behavior is achieved.

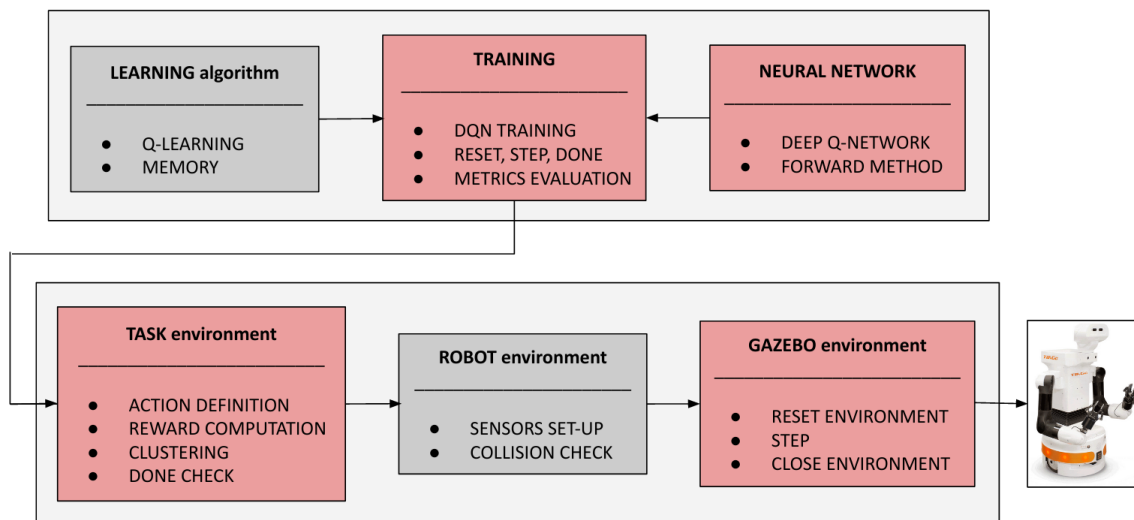


Figure 5.6: Scheme of the architecture to train active perception models with RL.

The network models chosen for both methods are more similar as possible, to allow a comparison as fair as possible. The only differences are due to suitable changes. In addition, these trained models can work directly on the robot. Once again, as data-driven methods, results improve with the increase of data relating to more scenarios, more runs, and more contexts of use.

In summary, the three contributions collectively form the basis of this research, providing novel insights and solutions for enhancing perception.

5.5 Clustering analysis for points grouping

Before proceeding with the mathematical definition of the problem to face, an optimal clustering algorithm to group points must be found. In this section, the clustering comparison analysis is reported.

The comparison takes into account three types of clusterings, with the analysis done for three sets of points, assuming that each point is a person⁴. All these people must be grouped. The target clustering method must be able to:

- Form groups with size ≥ 1 ;
- Enforce a minimum distance between points in different clusters, in a way that two points must be assigned to different clusters if their distance is greater than or equal to a specified minimum distance.

The value for this minimum distance has been chosen by referring to the Hall space model (Figure 2.3 Chapter 2): since the personal space of a person includes the space inside 1.2 m, two people are not interacting until their distance becomes ≥ 2.4 m (Figure 5.7).

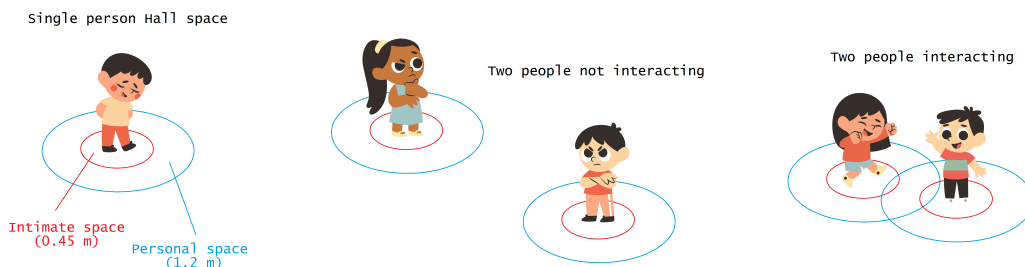


Figure 5.7: By considering the measures of the intimate and personal space as on the left of the figure according to Hall space convention, two people can be considered as interacting if their distance is ≤ 2.4 m, as in the right of the figure. Indeed, as soon as this distance increases, they are enough far away to belong to different social groups, hence not interacting (middle of the figure).

The clustering algorithms⁵ tested in this thesis are to compare are the Density-Based Spatial Clustering of Applications with Noise (DBSCAN), the Ordering Points To Identify Cluster Structure (OPTICS) and the Hierarchical Agglomerative. These clusterings have been chosen since they are the most suited to cluster data in groups in a similar way people do. The first test done with these

⁴This assumption is used here to simplify the problem to find an optimal clustering. Further, this will be applied to the set of points labeled as *LIKELY* and to those labeled as *UNDEFINED*.

⁵For further details: <https://scikit-learn.org/stable/modules/clustering.html>.

methods showed that hierarchical clustering returns always the desired output. Hence, DBSCAN and OPTICS are here compared with the hierarchical, handled as ground truth.

For each set of points, all the following elements are included:

- The plots of points and the clusterings outputs;
- The comparison DBSCAN vs hierarchical with a confusion matrix, indices histogram and table;
- The comparison OPTICS vs hierarchical with a confusion matrix, indices histogram and table.

The metrics computed for the comparison are:

- Adjusted Rand Index (ARI);
- Normalized Mutual Information (NMI);
- Fowlkes-Mallows index (FMI);
- Homogeneity;
- Completeness;
- V-Measure;
- Purity.

5.5.1 Preliminary tests on clustering

We tested three particular configurations of points to have a preliminary test about the correct functioning of the classical algorithms: DBSCAN, OPTICS, and hierarchical clustering. The configuration of points and the outputs of these algorithms are analyzed in Section 5.5.2, Section 5.5.3, and Section 5.5.4.

5.5.2 First experimented configuration

In Figure 5.8 there are the four plots of data and the clusterings outputs. In Figure 5.9 and in Figure 5.10 the hierarchical clustering is compared respectively to DBSCAN and OPTICS. Table 5.1 collects the metrics values to summarize what has been computed.

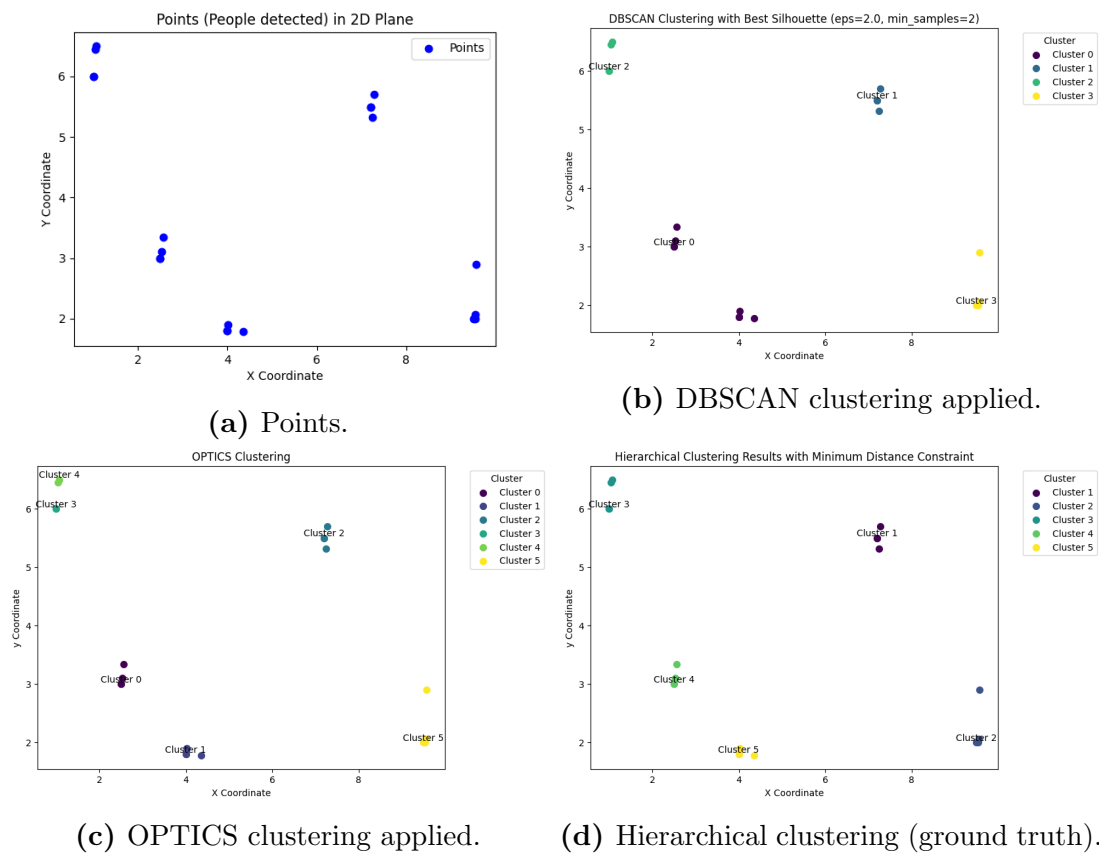


Figure 5.8: Plots of the first experimented configuration with the outputs of the clusterings.

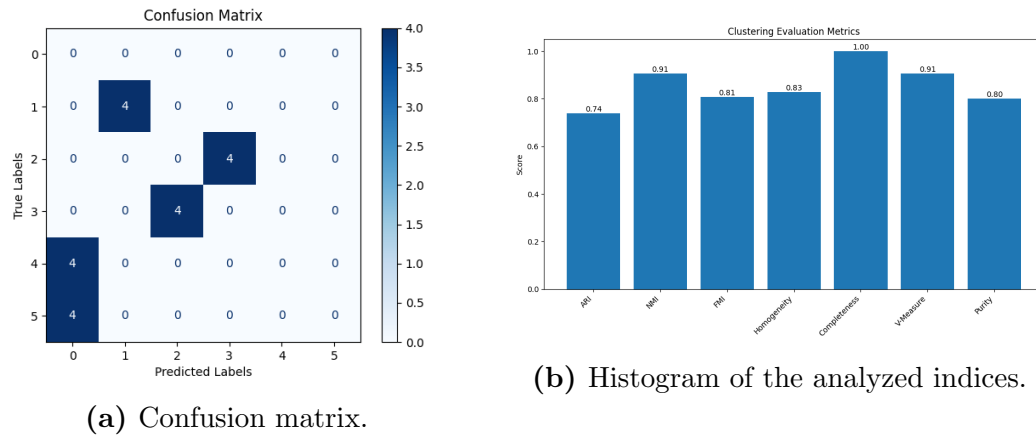


Figure 5.9: Metrics comparison between labels predicted by DBSCAN vs those of hierarchical clustering (ground truth labels).

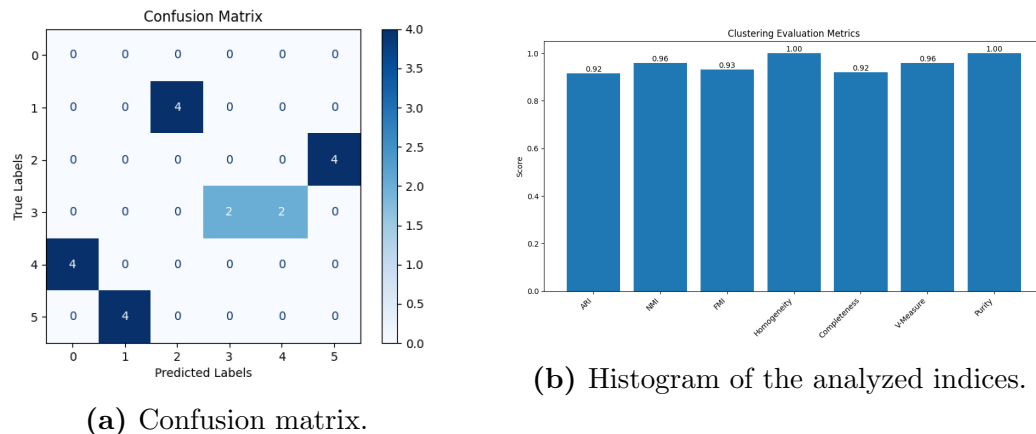


Figure 5.10: Metrics comparison between labels predicted by OPTICS vs those of hierarchical clustering (ground truth labels).

Metric	DBSCAN vs Hierarchical	OPTICS vs Hierarchical
ARI	0.73972	0.91629
NMI	0.90574	0.95871
FMI	0.80757	0.93094
Homogeneity	0.82772	0.99999
Completeness	1.00000	0.92069
V-Measure	0.90574	0.95871
Purity	0.80000	1.00000

Table 5.1: Table of metrics displayed in the histograms in Figure 5.9 and in Figure 5.10.

5.5.3 Second experimented configuration

In Figure 5.11 there are the four plots of data and the clusterings outputs. In Figure 5.12 and in Figure 5.13 the hierarchical clustering is compared respectively to DBSCAN and OPTICS. Table 5.2 collects the metrics values to summarize what has been computed.

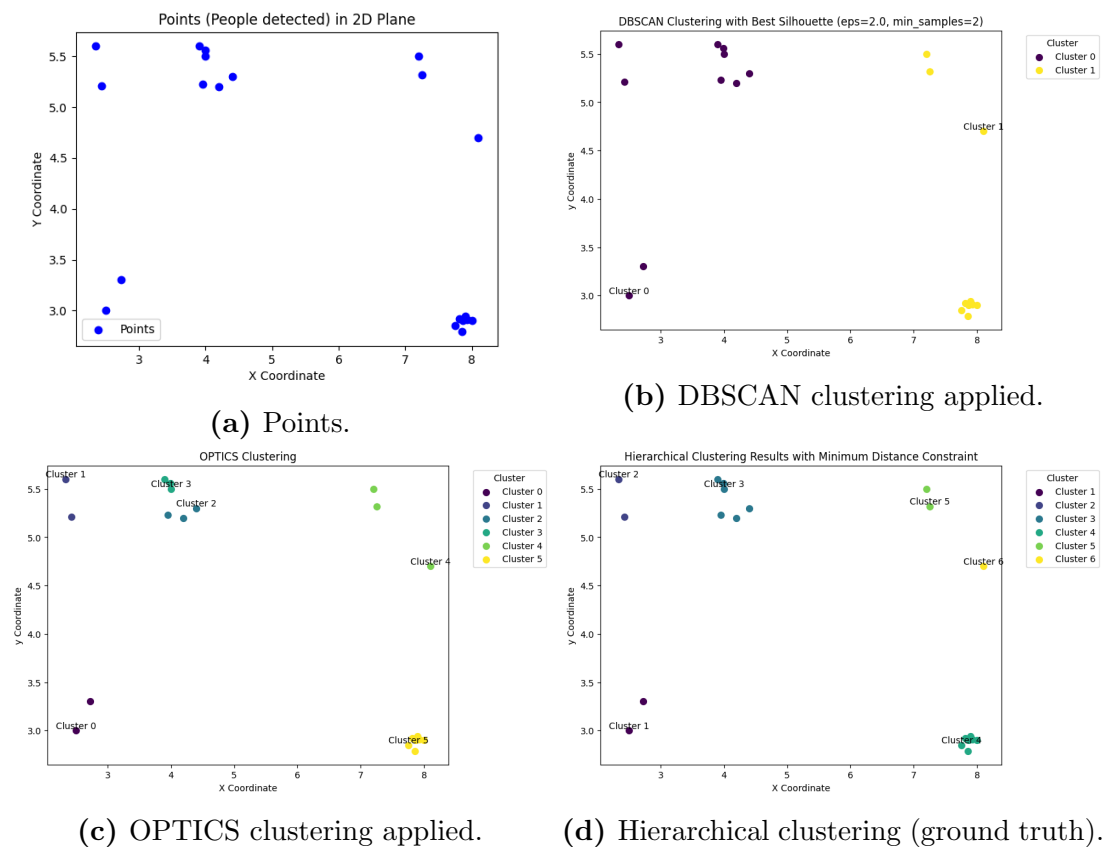
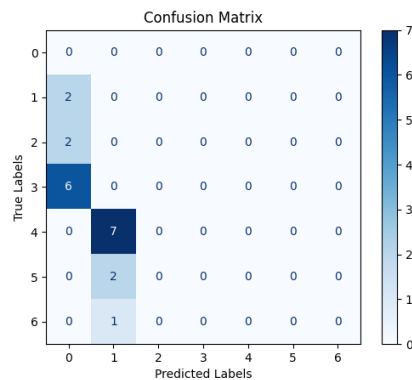
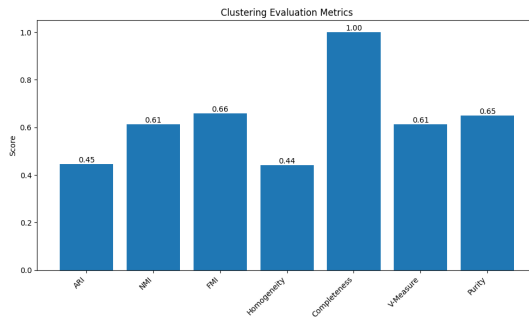


Figure 5.11: Plots of the second experimented configuration with the outputs of the clusterings.

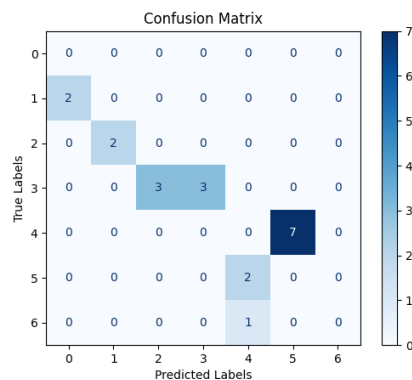


(a) Confusion matrix.

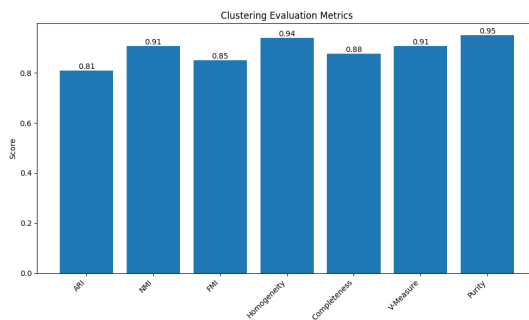


(b) Histogram of the analyzed indices.

Figure 5.12: Metrics comparison between labels predicted by DBSCAN vs those of hierarchical clustering (ground truth labels).



(a) Confusion matrix.



(b) Histogram of the analyzed indices.

Figure 5.13: Metrics comparison between labels predicted by OPTICS vs those of hierarchical clustering (ground truth labels).

Metric	DBSCAN vs Hierarchical	OPTICS vs Hierarchical
ARI	0.44597	0.80990
NMI	0.61277	0.90666
FMI	0.65828	0.84921
Homogeneity	0.44172	0.93916
Completeness	1.00000	0.87635
V-Measure	0.61277	0.90666
Purity	0.65000	0.95000

Table 5.2: Table of metrics displayed in the histograms in Figure 5.12 and in Figure 5.13.

5.5.4 Third experimented configuration

In Figure 5.14 there are the four plots of data and the clusterings outputs. In Figure 5.15 and in Figure 5.16 the hierarchical clustering is compared respectively to DBSCAN and OPTICS. Table 5.3 collects the metrics values to summarize what has been computed.

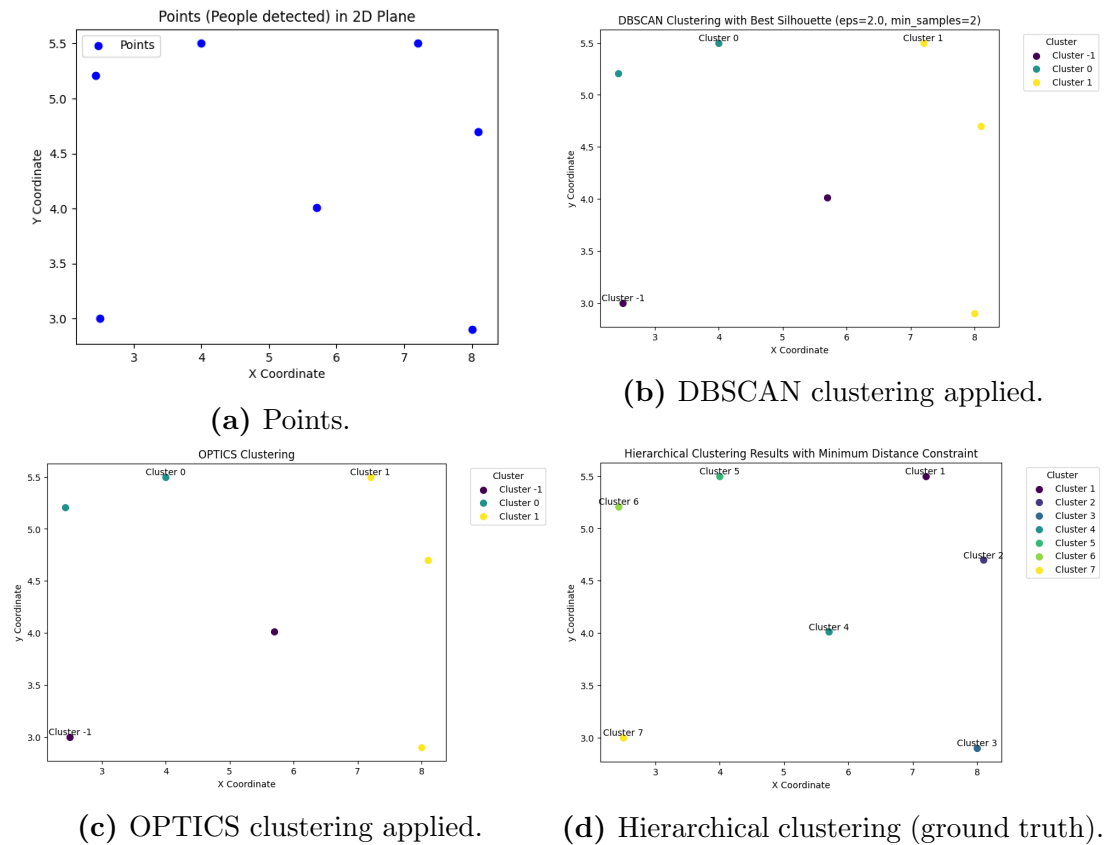
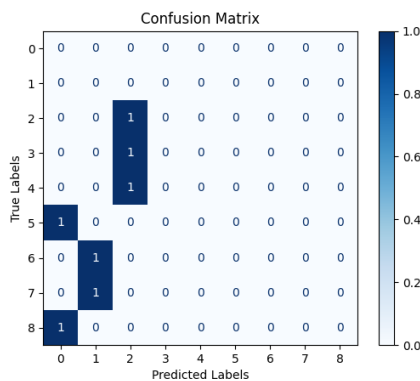


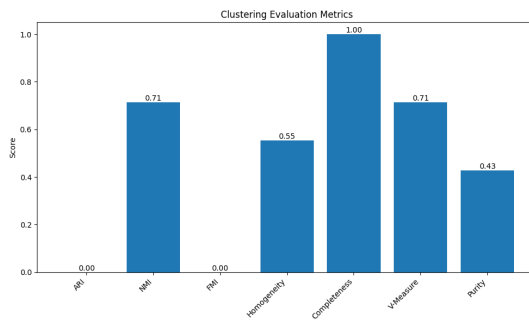
Figure 5.14: Plots of the third experimented configuration with the outputs of the clusterings.

Metric	DBSCAN vs Hierarchical	OPTICS vs Hierarchical
ARI	0.00000	0.00000
NMI	0.71341	0.71341
FMI	0.00000	0.00000
Homogeneity	0.55450	0.55450
Completeness	1.00000	1.00000
V-Measure	0.71341	0.71341
Purity	0.42857	0.42857

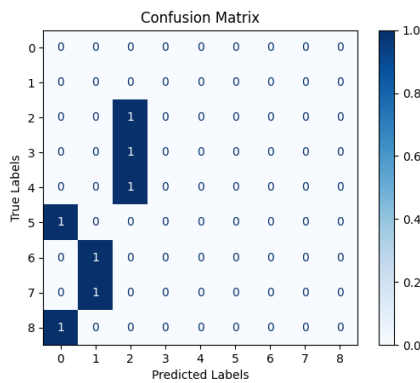
Table 5.3: Table of metrics displayed in the histograms in Figure 5.15 and in Figure 5.16.



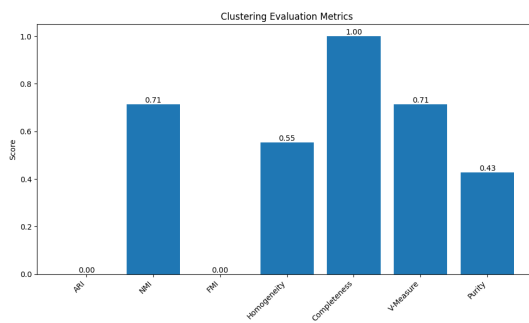
(a) Confusion matrix.



(b) Histogram of the analyzed indices.

Figure 5.15: Metrics comparison between labels predicted by DBSCAN vs those of hierarchical clustering (ground truth labels).

(a) Confusion matrix.



(b) Histogram of the analyzed indices.

Figure 5.16: Metrics comparison between labels predicted by OPTICS vs those of hierarchical clustering (ground truth labels).

5.5.5 Clusterings comparison conclusions

The plots and the metrics show that DBSCAN works worse than OPTICS, which works worse than hierarchical clustering. Both need some hyperparameters to set that, to get the best, have been ran with ranges of parameters. In this way, the best cluster is returned and displayed in this section.

In addition, both DBSCAN and OPTICS have been shown to produce clusters of size ≥ 2 , which is not true in the case of groups of people. Indeed, people standing or walking alone belong to a group that has a size equal to 1.

For these reasons, from this point the chosen clustering method is hierarchical agglomerative with an adjusted threshold (2.4 m), Ward⁶ linkage.

⁶Ward linkage criteria minimizes the sum of squared differences within all clusters. It is a variance-minimizing approach tackled with an agglomerative hierarchical approach.

5.6 Mathematical formulation of the problem

In this section the mathematical formulation of the problem to come to a proper policy is reported with the related elements: the possible actions for the robot, states, both the proposed utility and reward functions and other information related to the context of use.

5.6.1 Elements formalization

In the following, all primary problem elements are formalized:

- *Scenarios*: two types of scenarios are considered to apply this work. In the first (*DANGER*), the goal of the robot is to look for people in less possible time to save as many people as it can, while in the second one (*DISCOVER*) the robot has to discover as many people as possible⁷;

$$s(x) = \begin{cases} a_P = [1.0, 0.5, 0.1], & \text{if } x = \text{“DANGER”} \\ a_P = [0.5, 1.0, 0.1], & \text{if } x = \text{“DISCOVER”} \end{cases} \quad (5.8)$$

- *State*: the current state s_i for step i is represented by a vector of size 212×3 (please see Equation 5.6.1). The basic elements are:
 - Detection: vector of $n(x, y)$ points of the people detection, all with respect to a global frame;
 - Undefined: vector of $m(x, y)$ points labeled to belong to the *UNDEFINED* class (Section 4.3), also these with respect to a global frame;
 - Padding points: points that contain values of invalid meaning (e.g.: in a world of size $150m \times 270m$ a padding point can be $[500, 500]$);
 - Filter vectors: vector made up of 1 and 0 where 1 marks the point as valid, both for the detection and the undefined vectors.

An example of the final general state vector is above. The length of 106 for each point vector has been chosen to cover all detection cases, without losing detection in any situation. This is the reason why the length must approximate the maximum number of people the robot can detect. The

⁷In the *DISCOVER* scenario the time is important, but less important than in a dangerous scenario (e.g.: a fire).

$$\begin{aligned}
& \left[\begin{array}{l} \left[\begin{array}{l} x_{D1}, \quad y_{D1}, \quad f_{D1} = 1 \end{array} \right], \\ \left[\begin{array}{l} x_{D2}, \quad y_{D2}, \quad f_{D2} = 1 \end{array} \right], \\ \left[\begin{array}{l} x_{D\dots}, \quad y_{D\dots}, \quad f_{D\dots} = 1 \end{array} \right], \\ \left[\begin{array}{l} x_{Dn}, \quad y_{Dn}, \quad f_{Dn} = 1 \end{array} \right], \\ \left[\begin{array}{l} x_{Dn+1}, \quad y_{Dn+1}, \quad f_{Dn+1} = 0 \end{array} \right], \\ \left[\begin{array}{l} x_{D\dots}, \quad y_{D\dots}, \quad f_{D\dots} = 0 \end{array} \right], \\ \left[\begin{array}{l} x_{D106}, \quad y_{D106}, \quad f_{D106} = 0 \end{array} \right], \\ \left[\begin{array}{l} x_{U1}, \quad y_{U1}, \quad f_{U1} = 1 \end{array} \right], \\ \left[\begin{array}{l} x_{U2}, \quad y_{U2}, \quad f_{U2} = 1 \end{array} \right], \\ \left[\begin{array}{l} x_{U\dots}, \quad y_{U\dots}, \quad f_{U\dots} = 1 \end{array} \right], \\ \left[\begin{array}{l} x_{Um}, \quad y_{Um}, \quad f_{Um} = 1 \end{array} \right], \\ \left[\begin{array}{l} x_{Um+1}, \quad y_{Um+1}, \quad f_{Um+1} = 0 \end{array} \right], \\ \left[\begin{array}{l} x_{U\dots}, \quad y_{U\dots}, \quad f_{U\dots} = 0 \end{array} \right], \\ \left[\begin{array}{l} x_{U106}, \quad y_{U106}, \quad f_{U106} = 0 \end{array} \right] \end{array} \right]
\end{aligned}$$

Table 5.4: General state vector for RL training.

robot used in this work is the TIAGo base, so the length computation started from the laser information.

To maximize the number, occlusion can not occur, hence there is no person in front of another. Hence, they must be one next to the other, at the same distance from the robot. The bigger this distance, the bigger the number of people in the scene. This distance is the maximum radius of the laser (range_max [m]), as in Figure 5.17. In that figure, there are only eleven people, but the idea is that one must stay next to the other while respecting at limit the intimate area of the Hall space (radius of 0.45 m). Thus, the approximate number of people becomes 106. The state, in this problem, is the state of the perceived environment;

$$\begin{aligned}
\frac{\text{length_circular_sector}}{\text{diameter_intimate_space}} &= \frac{(\text{angle_max} - \text{angle_min})/360^\circ \times 2 \times \pi \times \text{range_max}}{2 \times 0.45\text{m}} \\
&= \frac{(110^\circ - (-110^\circ))/360^\circ \times 2 \times \pi \times 25\text{m}}{2 \times 0.45\text{m}} \simeq 106 \quad (5.9)
\end{aligned}$$

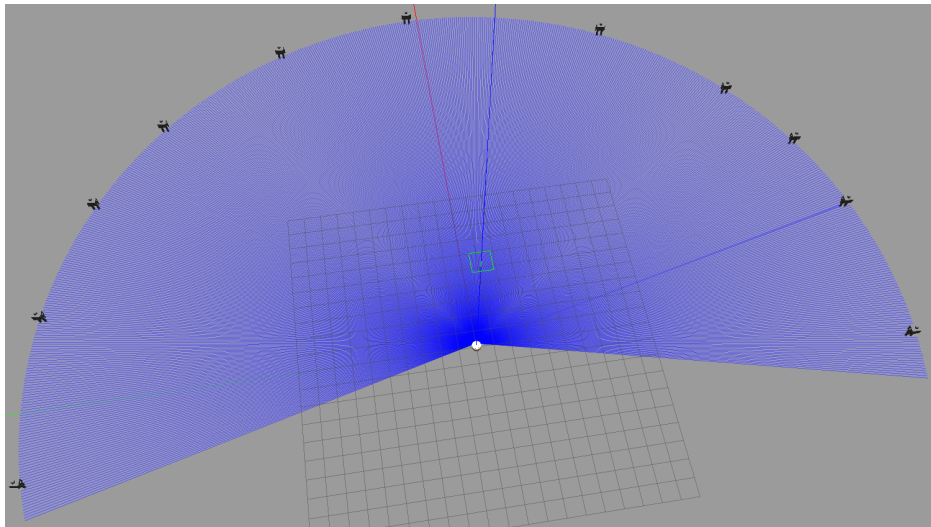


Figure 5.17: Example of people configuration to maximize the number of detection (106).

```
1 ---
2 seq: 162
3 stamp:
4 secs: 564
5 nsecs: 602000000
6 frame_id: "base_laser_link"
7 angle_min: -1.80437970161
8 angle_max: 1.80437970161
9 angle_increment: 0.00577401509508
10 time_increment: 0.0
11 scan_time: 0.0
12 range_min: 0.0500000007451
13 range_max: 25.0
14 ranges: [0.11099119484424591, 0.1106792539358139,
           0.11036767810583115, 0.11013428866863251, ...,
           0.11143097281455994, 0.11165988445281982,
           0.11196549981832504, 0.11196549981832504]
15 intensities: [0.0, 0.0, 0.0, 0.0, ..., 0.0, 0.0, 0.0, 0.0]
16 ---
```

Listing 5.1: Laser scan message example

- *Actions*: $a = \{ \text{GO LIKELY}, \text{GO UNDEFINED}, \text{SCANNING} \}$. In the code the set of possible actions will be codified as $a = \{ 0, 1, 2 \}$;
- *Actions Priorities*: $a_P = \{ \alpha_{a_P}, \beta_{a_P}, \gamma_{a_P} \}$ are the general priority weights. According to the scenario of use, these weights assume the values in Equation $s(x)$;
- *Reward function*: at each step t , the reward must be updated as in Equation 5.10 by adding to the current reward the one computed for the specific step. A significant reward has to be found, better if with a meaning for the problem, both as a positive case and also as a penalty (Section 5.6.4). The main point for this purpose is to define the so-called $\Delta R_{t,t-1}$ to gain after an action between two consecutive steps;

$$R_t = R_{t-1} + \Delta R_{t,t-1}, \quad R_{t-1} = \sum_{t'=1}^{t-1} \Delta R_{t',t'-1} \quad (5.10)$$

- *Clustering method*: given the points above, a perfect clustering method is essential to cluster points in the best way to catch potential social groups of people (Section 5.5);
- *Group utility function*: for each group of the clustering obtained as the output of the clustering method, applied to both sets of labeled points, a group utility score must be computed. The bigger this index, the more it should reflect the utility for the robot to approach that group, by weighting context data (Section 5.6.2);
- *ID assignment*: given a set of new points just collected $IDS = \{(ID_1, p_1), (ID_2, p_2), \dots, (ID_K, p_K)\}$ where $|IDS| = K$ and $p_i = (x_i, y_i) \forall 1 \leq i \leq K$, an *ID* to each of those points must be assigned to keep track of them in the future. As soon as a new point $p_i = (x_i, y_i)$ is acquired, it is coupled with the first integer not assigned yet⁸, then inserted in the list;
- *Memory list*: in this list, all visited points are recorded as couples (*key*, *value*) = (*ID*, *point*), so that in the case of a future action a given point $p_i = (x_i, y_i)$ will be visited again, it will be remembered if a couple of the type $(key_{p_i}, value_{p_i}) = (ID_{p_i}, p_i)$ is already present in this memory list;

⁸To ensure the key value key_{p_i} is a monotonically increasing function, avoiding duplicates.

- *Working policy*: given a current context (iteration), the robot must follow the high-level steps of Algorithm 15.

Algorithm 15 Robot Working Policy

Input: LiDAR sensor data at time t

Output: Current iteration reward R_{t+1}

- 1:
 - 2: Compute sensor data classification and people detection (Section 4.3);
 - 3: Save the number of people detected;
 - 4: Update list of *IDs* and memory list;
 - 5:
 - 6: Run clustering on likely and undefined points;
 - 7: Compute utility score for each group in likely points clustering;
 - 8: Get the maximum of likely scores;
 - 9: Compute utility score for each group in undefined points clustering;
 - 10: Get the maximum of undefined scores;
 - 11: Compute utility score for the scanning action;
 - 12: Get the maximum among all max utility indices to choose the action;
 - 13: Execute the chosen action;
 - 14:
 - 15: Detect people again and save the number of people detected;
 - 16: Compute the difference in the number of people detected before and after the action;
 - 17: Compute the current iteration reward.
 - 18:
- return** Reward R_{t+1} ;
-

5.6.2 Design of the problem

To design the problem, precise meaning must be provided for the possible actions. In the next paragraphs, we will present the definition of the proposed actions: Go Likely and Go Undefined, and Scanning. Finally, we specify the action selection.

Go Likely and Go Undefined

Starting from the laser scan topics where points labeled as likely and undefined are stored, the two sets of points $points_{likely}$ and $points_{undefined}$ can be built. The same holds for the positions of people detected, available in the proper topic after publication done by the people detector (Section 4.3).

Through clustering, points are divided into groups. For each group of each set, the already mentioned utility score must be produced.

This function depends on the current distance robot-group called d and on the group size. In general, points in a group can have been already visited. For this reason, the list of IDs assigned to points and the memory list helps to keep track of those visited. This is the reason why the second parameter for the utility function is the number x of points in a group that, once visited, can increase the length of the memory list. Indeed, in the case where no point has been visited, this number coincides with the size of the group, while if only a few are not in the memory list, then the group must get less importance. Since just in case of a group completely visited x is equal to zero, a little ϵ^9 must be added properly. In addition, in case the increase in length of the memory list is zero the function should return a penalty, otherwise a monotonically increasing positive number¹⁰.

Taking into account this information, the first version of the utility function is in Equation 5.11.

$$U(x, d, \epsilon) = \begin{cases} \ln(x + \epsilon) & \text{if } 0 \leq x \leq 0.5 \\ \frac{e^x}{d^2} & \text{if } x > 0.5 \end{cases} \quad (5.11)$$

Then, these two pieces of the previous equation can be joined into Equation 5.12. Figure 5.18 shows the graphic plots of these equations.

$$U(x, d, \epsilon) = \frac{e^x}{d^2} + \ln(x + \epsilon), \quad x \in \{0\} \cup \mathbb{N} \quad (5.12)$$

The inclusion of the two functions in Equation 5.11 into a single function 5.12 is justified by the following reasons:

1. A single function is more appropriate than a piecewise function so that no particular cases must be analyzed separately but after the study of this section a single, general, and simpler function is obtained and it can be applied without taking into account again the particular value in input;
2. The comparison of a single function can be done by working only on the other parameters (e.g.: d, ϵ), without the need to specify individually each piece of the function;

⁹The proposed value for ϵ is 1e-15, but it can be modified at will, provided it is very little.

¹⁰The threshold number of Equation 5.11 must be a number $t \mid \epsilon < t < 1$. The proposed threshold is 0.5 .

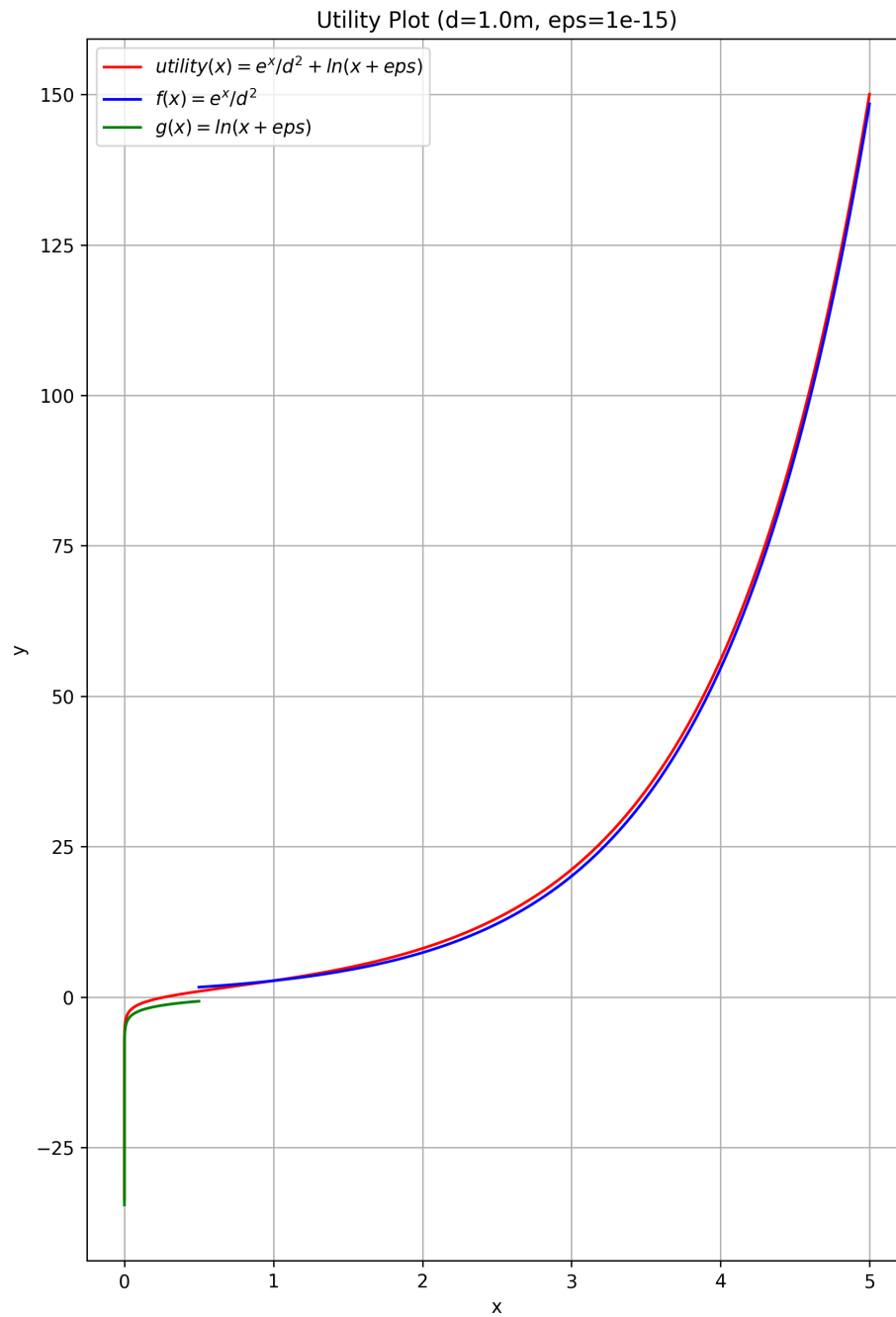


Figure 5.18: Simple plot of the group utility function ($d = 1.0m$, $\epsilon = 1e - 15$).

3. The difference between the functions defined in Equation 5.12 and in Equation 5.11 is almost zero in the whole domain, as can be seen for example in Figure 5.19.

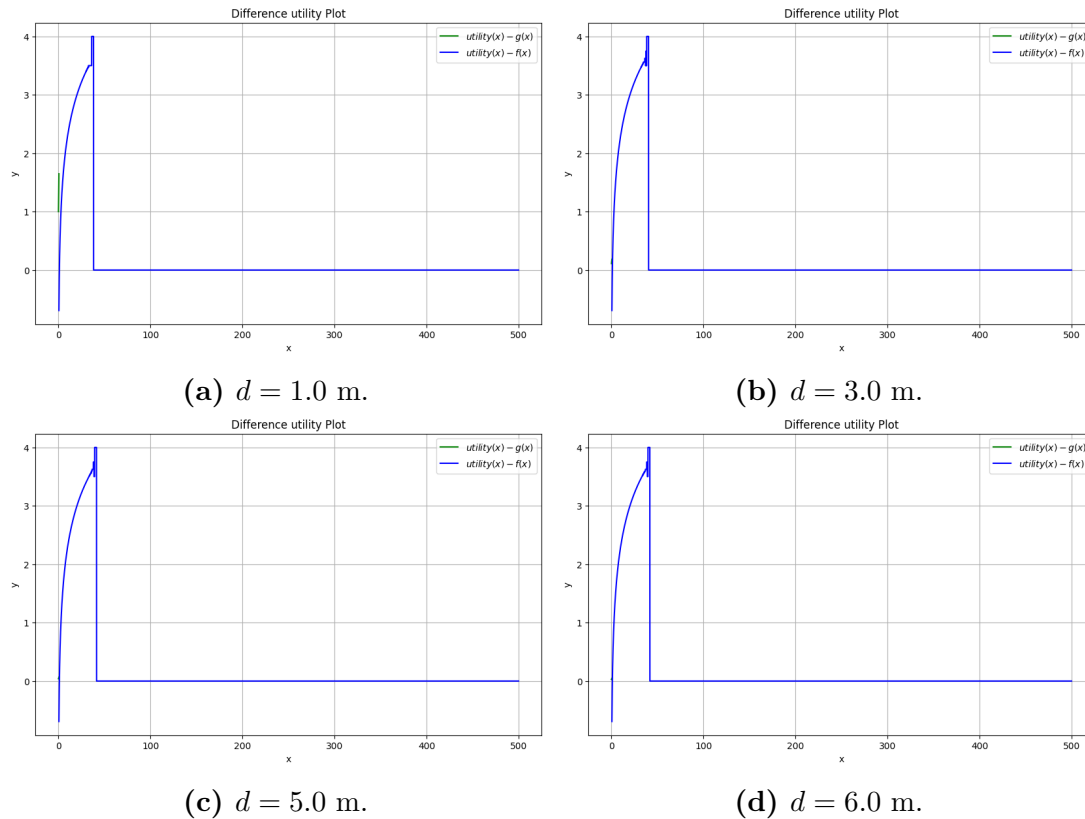


Figure 5.19: Plots of the difference between the group utility function and the piecewise one, for distance $d \in \{1, 3, 5, 6\}$ ($\epsilon = 1e - 15$).

As can be seen, the maximum difference is about 4 units (regardless of the d value) which does not affect significantly the work of the method. The influence of the d value on the function shape, instead, is analyzed in Figure 5.20 and in Figure 5.21. The focus of these two pictures is respectively on the left (small x values) and the right part (bigger x values) of the shape.

Scanning

In case there is no point belonging to the *LIKELY* class or to the *UNDEFINED* class, a particular action is needed. To do so, the *SCANNING* action helps to solve or at least fight this behavior.

Given the sets of likely (L) and undefined points (U), let's call $|L|$ and $|U|$ the cardinalities of these two sets, hence since $N = L \cup U$, $|N| = |L| + |U|$. The utility of this action must return a big number as soon as one of these two sets is going (or is/are) empty.

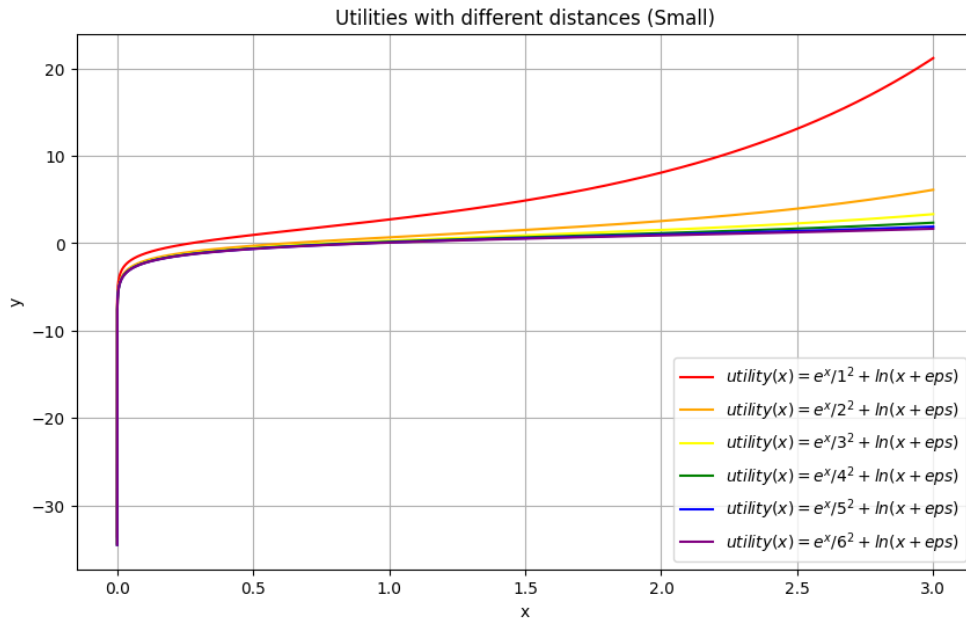


Figure 5.20: Small overview of the group utility functions with different distance measures (d [m]) and fixed $\epsilon = 1e - 15$. Here the focus is on the left side of the function (little input values).

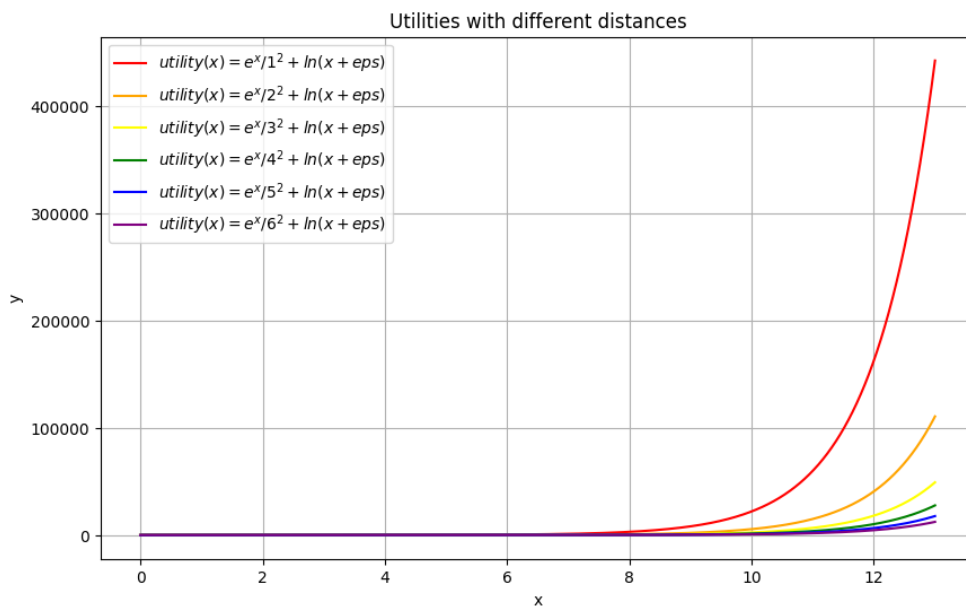


Figure 5.21: Overview of the group utility functions with different distance measures (d [m]) and fixed $\epsilon = 1e - 15$. Here the focus is on the right side of the function (bigger and more relevant input values).

$$U(L, U) = \frac{(|L| + |U|) \cdot (|L| + |U|)}{|L| \cdot |U|} = \frac{|N|^2}{|L| \cdot |U|} \quad (5.13)$$

To avoid divisions by zero, let's add two epsilons in the proper positions so that Equation 5.13 becomes Equation 5.14.

$$U(L, U) = \frac{(|L| + |U|) \cdot (|L| + |U|)}{(|L| + \epsilon) \cdot (|U| + \epsilon)} = \frac{|N|^2}{(|L| + \epsilon) \cdot (|U| + \epsilon)} \quad (5.14)$$

In this way, in case of $|L|$ or $|U|$ that tend to zero, $U(SCANNING)$ becomes a big number, that multiplied by $a_P[2]$ (e.g.: $a_P[2] = 0.1$), still is a big number that gives *SCANNING* action a high probability of being chosen among all possible actions.

As soon as the *SCANNING* action is selected, the robot is subject to a little random movement of position and rotation (pose) so that a new iteration can happen by doing a new scanning of the current situation.

Action selection

The final action will be the one that maximizes the utility score, weighted with the action priorities $a_P = \{ \alpha_{a_P}, \beta_{a_P}, \gamma_{a_P} \}$.

$$max_L = max\{U(x_{C_i}, d_{C_i}, \epsilon), \forall C_i \in Cluster(points_{likely})\} \quad (5.15)$$

$$max_U = max\{U(x_{C_j}, d_{C_j}, \epsilon), \forall C_j \in Cluster(points_{undefined})\} \quad (5.16)$$

$$max_S = U(L = points_{likely}, U = points_{undefined}) = \frac{|N|^2}{(|L| + \epsilon) \cdot (|U| + \epsilon)} \quad (5.17)$$

$$action = \operatorname{argmax}_{a_P \cdot U} [a_P[0] \cdot max_L, a_P[1] \cdot max_U, a_P[2] \cdot max_S] \quad (5.18)$$

5.6.3 Example of algorithm executions

In this section, one example of execution is displayed for each action (go likely, go undefined, scanning). For instance, we examine such actions in the *DISCOVER* scenario to have a coherent comparison.

GO LIKELY

Given the current context in Figure 5.22, after the clustering of group points, the utility score must be computed for each group.



Figure 5.22: Example of a starting scenario. As shown in the legend, the circles represent the points labeled as LIKELY, the crosses stand for those UNDEFINED, and the grey rhombus is placed in the robot position.

The utilities computed for both sets of points are in Table 5.5 and Table 5.6. Also, the utility for the scanning action is calculated with $|L| = 10$, $|U| = 10$, $\epsilon = 1e - 15$.

Group LIKELY	x	d	e^x/d^2	$\ln(x + \epsilon)$	$U(x, d, \epsilon)$
Cluster 1	2	1.26025	4.65240	0.69315	5.34555
Cluster 2	2	1.21748	4.98503	0.69315	5.67818
Cluster 3	6	2.37010	71.81796	1.79176	73.60972

Table 5.5: Table of indexes computed for all groups in the LIKELY clustering. The bold cluster is the one with maximum utility for the potential Go Likely action.

Group UNDEFINED	x	d	e^x/d^2	$\ln(x + \epsilon)$	$U(x, d, \epsilon)$
Cluster 1	7	6.02892	30.17053	1.94591	32.11644
Cluster 2	2	5.35222	0.25794	0.69315	0.95109
Cluster 3	1	6.11658	0.07266	1.11022e-15	0.07266

Table 5.6: Table of indexes computed for all groups in the UNDEFINED clustering. The bold cluster is the one with maximum utility for the potential Go Undefined action.

The GO LIKELY action will be executed (Table 5.7): the robot will choose to get closer¹¹ to Group 3 (Figure 5.23), which mean position is at (4.07333, 5.39833) at a distance robot-group of 2.37010.

¹¹In these examples, the distance robot-group after the action is 1.2 m to simulate an approaching of the robot into the group's personal space (Hall model).

Action	Max utility	Action weight	Final utility
Go Likely	73.60972	0.5	36.80486
Go Undefined	32.11644	1.0	32.11644
Scanning	4.00000	0.1	0.40000

Table 5.7: Table of indexes computed for all potential actions, to decide which can have a higher utility. The bold action is the one that will be executed.

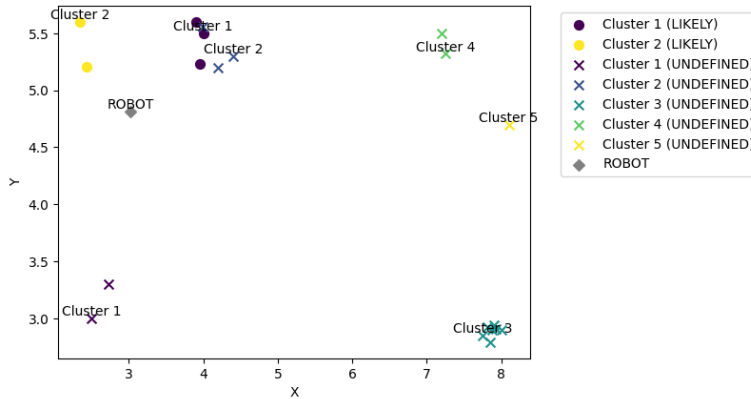


Figure 5.23: Scenario after the action. The robot is closer to the chosen group.

GO UNDEFINED

Given the current context in Figure 5.24, after the clustering of group points, the utility score must be computed for each group.

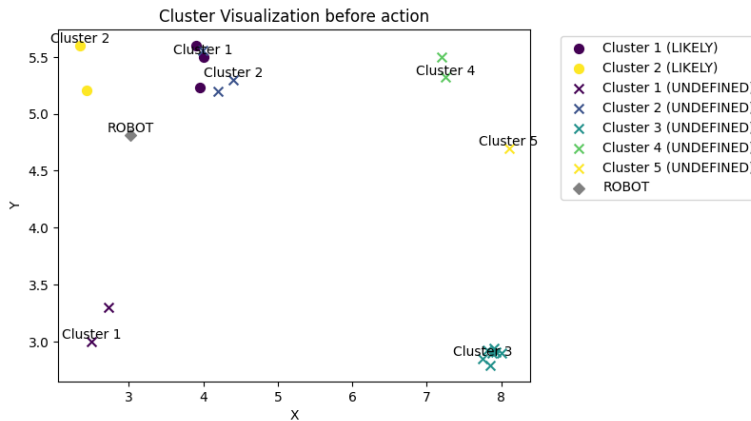


Figure 5.24: Example of a starting scenario. As shown in the legend, the circles represent the points labeled as LIKELY, the crosses stand for those UNDEFINED, and the grey rhombus is placed in the robot position.

The utilities computed for both sets of points are in Table 5.8 and Table 5.9. Also, the utility for the scanning action is calculated with $|L| = 5$, $|U| = 15$, $\epsilon = 1e - 15$.

Group LIKELY	x	d	e^x/d^2	$\ln(x + \epsilon)$	$U(x, d, \epsilon)$
Cluster 1	0	1.11831	0.79956	-34.53878	-33.73922
Cluster 2	2	0.86812	9.80462	0.69315	10.49777

Table 5.8: Table of indexes computed for all groups in the LIKELY clustering. The bold cluster is the one with maximum utility for the potential Go Likely action.

Group UNDEFINED	x	d	e^x/d^2	$\ln(x + \epsilon)$	$U(x, d, \epsilon)$
Cluster 1	2	1.71627	2.50852	0.69315	3.20167
Cluster 2	3	1.28990	12.07175	1.09861	13.17036
Cluster 3	7	5.21914	40.25900	1.94591	42.20491
Cluster 4	2	4.24306	0.41042	0.69315	1.10357
Cluster 5	1	5.07776	0.10543	1.11022e-15	0.10543

Table 5.9: Table of indexes computed for all groups in the UNDEFINED clustering. The bold cluster is the one with maximum utility for the potential Go Undefined action.

Action	Max utility	Action weight	Final utility
Go Likely	10.49777	0.5	5.24888
Go Undefined	42.20491	1.0	42.20491
Scanning	5.33333	0.1	0.53333

Table 5.10: Table of indexes computed for all potential actions, to decide which can have a higher utility. The bold action is the one that will be executed.

The GO UNDEFINED action will be executed (Table 5.10): the robot will choose to get closer to Group 3 (Figure 5.25), which mean position is at (7.87286, 2.88714) at a distance robot-group of 5.21914.

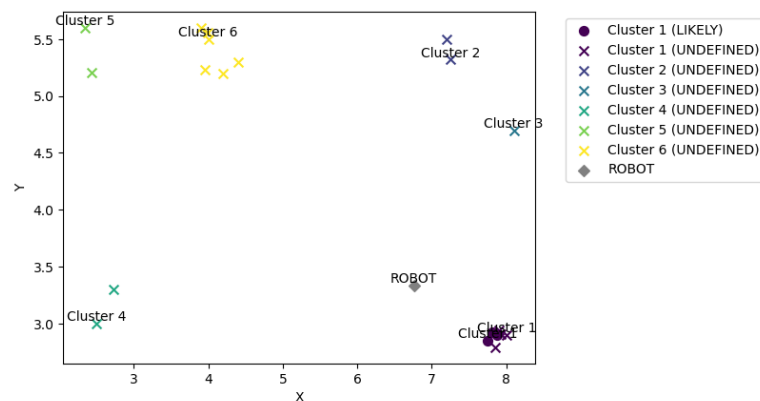


Figure 5.25: Scenario after the action. The robot is closer to the chosen group.

SCANNING

Given the current context in Figure 5.26, after the clustering of group points, the utility score must be computed for each group.

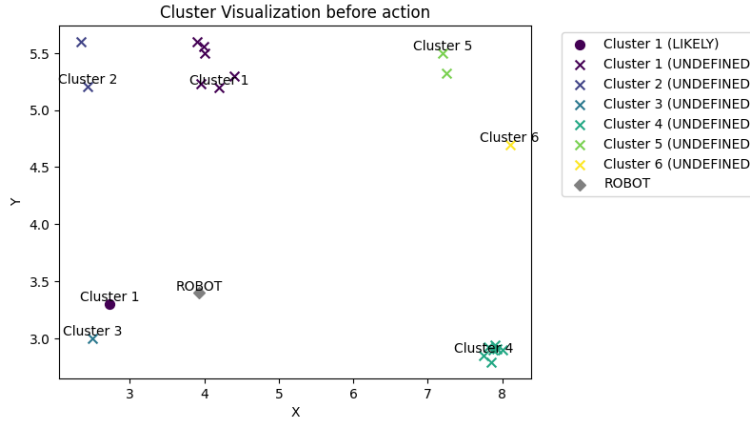


Figure 5.26: Example of a starting scenario. As shown in the legend, the circles represent the points labeled as LIKELY, the crosses stand for those UNDEFINED, and the grey rhombus is placed in the robot position.

The utilities computed for both sets of points are in Table 5.11 and Table 5.12. Also, the utility for the scanning action is calculated with $|L| = 1$, $|U| = 19$, $\epsilon = 1e - 15$.

Group LIKELY	x	d	e^x/d^2	$\ln(x + \epsilon)$	$U(x, d, \epsilon)$
Cluster 1	0	1.19928	0.69528	-34.53878	-33.84350

Table 5.11: Table of indexes computed for all groups in the LIKELY clustering. The bold cluster is the one with maximum utility for the potential Go Likely action.

Group UNDEFINED	x	d	e^x/d^2	$\ln(x + \epsilon)$	$U(x, d, \epsilon)$
Cluster 1	0	2.00785	0.24805	-34.53878	-34.29073
Cluster 2	0	2.53165	0.15602	-34.53878	-34.38276
Cluster 3	0	1.47941	0.45690	-34.53878	-34.08188
Cluster 4	1	3.98007	0.17160	1.11022e-15	0.17160
Cluster 5	1	3.86569	0.18190	1.11022e-15	0.18190
Cluster 6	1	4.37350	0.14211	1.11022e-15	0.14211

Table 5.12: Table of indexes computed for all groups in the UNDEFINED clustering. The bold cluster is the one with maximum utility for the potential Go Undefined action.

Action	Max utility	Action weight	Final utility
Go Likely	-33.84350	0.5	-16.92175
Go Undefined	0.18190	1.0	0.18190
Scanning	21.05263	0.1	2.10526

Table 5.13: Table of indexes computed for all potential actions, to decide which can have an higher utility. The bold action is the one that will be executed.

The SCANNING action will be executed (Table 5.13): the robot will go in a random position at a distance of 0.5 m from its previous one¹² (Figure 5.27). Here the position value changes from (3.92544, 3.39594) to (3.82113, 3.86131).

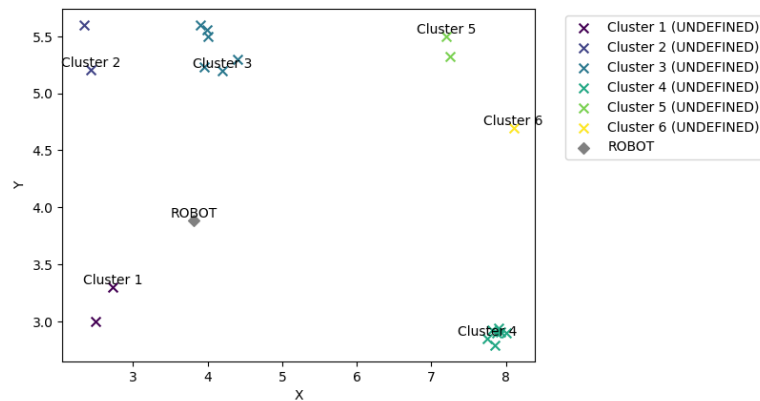


Figure 5.27: Scenario after the action. The robot has moved away from the previous position with a random movement.

5.6.4 Reward function

Starting from two consecutive detection, a general idea could be to compute a simple reward that counts the difference $\Delta p_{t,t-1}$ between detection from step t as compared to step $t - 1$. This is too simple and it would not work when an agent is completing the detection and only a few people are missing, so it would continuously get a reward very close to zero, classifying its work to be of poor quality, but it is not true.

This is the reason why a more complete and suitable reward function is required for this problem that has to reflect a balancing of these important features:

- *Difference* $\Delta p_{t,t-1}$: the reward has to reflect both the case of gain in detection and of loss in detection;

¹²In this example, the distance of robot motion after the SCANNING action is 0.5 m to simulate a little random movement, trying to move from local minimum positions.

- *Common detection* $p_{common_{t,t-1}}$: the reward has to increase according to the number of people already detected and still confirmed by the following detection;
- *New detection* $p_{new_{t,t-1}}$: in the case in which there is a gain in detection, the reward must be influenced also by the number of people detected at step t that had not been detected at step $t - 1$;
- *Lost detection* $p_{lost_{t,t-1}}$: in the case in which there is a loss in detection, the reward must be influenced also by the number of people detected at step $t - 1$ that have not been detected also at step t .

The function in Equation 5.19 must change according to the scenario of use: a parametric function must be modeled (Equation 5.20).

$$\Delta R_{t,t-1} = \begin{cases} \Delta p_{t,t-1} + p_{common_{t,t-1}} + p_{new_{t,t-1}}, & \text{if } \Delta p_{t,t-1} > 0 \\ \Delta p_{t,t-1} + p_{common_{t,t-1}} - p_{lost_{t,t-1}}, & \text{if } \Delta p_{t,t-1} \leq 0 \end{cases} \quad (5.19)$$

Table 5.14 collects the sign analysis of the components of the function.

$p_{common_{t,t-1}}$	$p_{new_{t,t-1}}$	$p_{lost_{t,t-1}}$	$\Delta R_{t,t-1}$
$\Delta p_{t,t-1} > 0$	≥ 0	> 0	> 0
$\Delta p_{t,t-1} \leq 0$	≥ 0	≥ 0	$\leq 0, = 0, \geq 0$

Table 5.14: Sign analysis of the reward function components.

$$\Delta R_{t,t-1} = \begin{cases} \alpha_{1,1}\Delta p_{t,t-1} + \beta_{1,2}p_{common_{t,t-1}} + \gamma_{1,3}p_{new_{t,t-1}}, & \text{if } \Delta p_{t,t-1} > 0 \\ \alpha_{2,1}\Delta p_{t,t-1} + \beta_{2,2}p_{common_{t,t-1}} + \gamma_{2,3}p_{lost_{t,t-1}}, & \text{if } \Delta p_{t,t-1} \leq 0 \end{cases} \quad (5.20)$$

A more compact notation can be used to collect these parameters:

- *Detection array*: it collects detection data as in Equation 5.21;

$$[\Delta p_{t,t-1}, p_{common_{t,t-1}}, p_{new_{t,t-1}}]^T \quad (5.21)$$

- Reward weights matrix: in the first row there are the parameters for the $\Delta p_{t,t-1} > 0$ case, while the second one is related to the $\Delta p_{t,t-1} \leq 0$ case (Equation 5.22).

$$\begin{pmatrix} \alpha_{1,1} & \beta_{1,2} & \gamma_{1,3} \\ \alpha_{2,1} & \beta_{2,2} & \gamma_{2,3} \end{pmatrix} \quad (5.22)$$

Since there can be two types of scenarios, these values must be set properly to capture the scenario features. The proposed reward weights are those in the Equation 5.23 for the *DISCOVER* scenario, while Equation 5.24 is for the *DANGER* one. Respectively, the former scenario focuses on new people to find, hence the loss of past detection has less importance than the discovery of new people; while in the latter scenario, the robot has mainly to move to collect as many detection as possible, hence gain detection is highly rewarded while the loss in detection is highly penalized.

$$\begin{pmatrix} 1 & 0.5 & 2 \\ 1 & 0.5 & -0.5 \end{pmatrix} \quad (5.23)$$

$$\begin{pmatrix} 1 & 1 & 2 \\ 1 & 1 & -2 \end{pmatrix} \quad (5.24)$$

As a result, the final reward function can be represented by the matrix-vector multiplication (Equation 5.25). The final reward value, instead, is the first component for the gain in detection when the number of people detected increases, otherwise the second one.

$$\begin{pmatrix} \alpha_{1,1} & \beta_{1,2} & \gamma_{1,3} \\ \alpha_{2,1} & \beta_{2,2} & \gamma_{2,3} \end{pmatrix} \begin{pmatrix} \Delta p_{t,t-1} \\ p_{common,t,t-1} \\ p_{new,t,t-1} \end{pmatrix} = \begin{pmatrix} reward_{gain\ detection} \\ reward_{loss\ detection} \end{pmatrix} \quad (5.25)$$

5.6.5 Example of reward function application

In this section, one simple example of the application of the reward function is presented for the two possible cases. The scenario for these examples is DISCOVER.

Gain in people detection

In Figure 5.28 the meaningful detection to compute $R_{t,t-1}$ are shown. A more comprehensible view is in Figure 5.30. The final values of the components defined to compute the reward function are in Table 5.15.

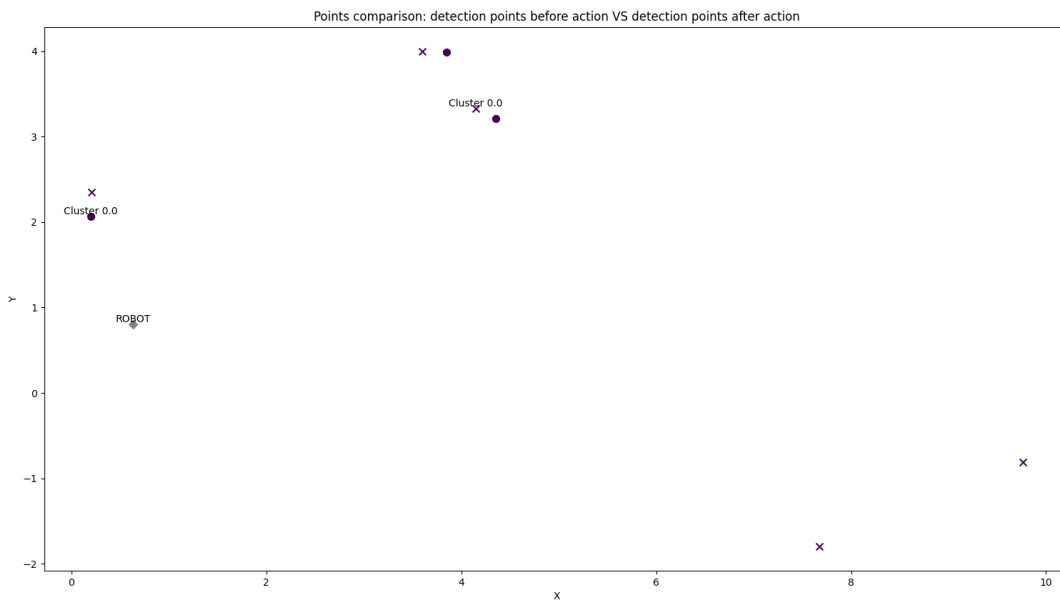


Figure 5.28: Gain in detection: plot of the detection before (the purple dots) and after (the purple crosses) step t . Dots closer enough to crosses are tracked to be the same person.

Component	Value
$p_{t-1,t-2}$	3
$p_{t,t-1}$	5
$\Delta p_{t,t-1}$	+2
$p_{common,t,t-1}$	3
$p_{new,t,t-1}$	2
$p_{lost,t,t-1}$	0
$\Delta R_{t,t-1}$	+7.5

Table 5.15: Gain in detection: analysis of the reward function components.

Loss in people detection

In Figure 5.29 the meaningful detection to compute $R_{t,t-1}$ are shown. A more comprehensible view is in Figure 5.31. The final values of the components defined to compute the reward function are in Table 5.16.

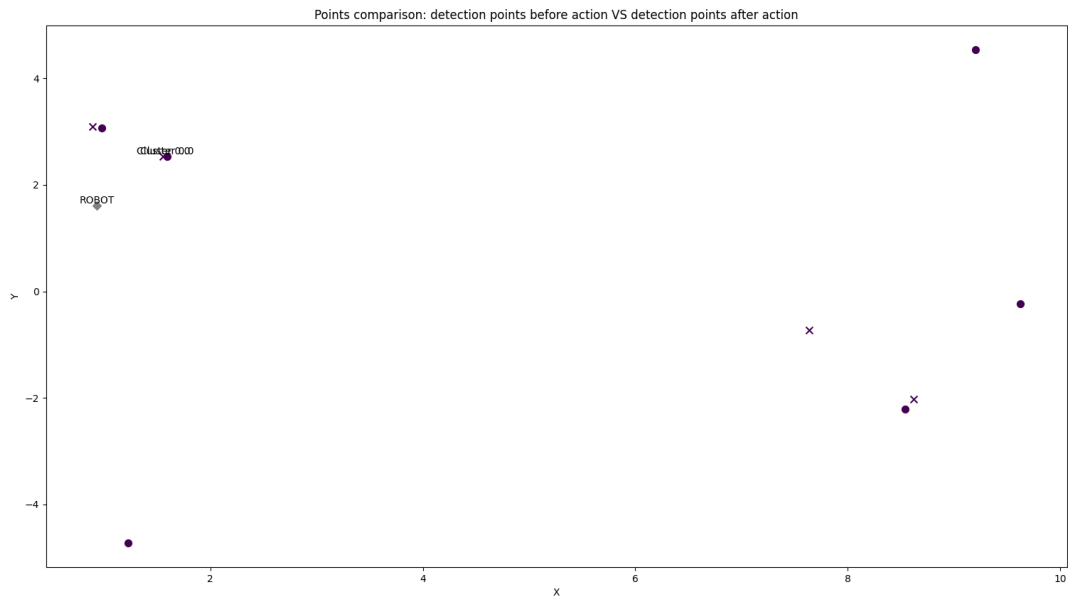


Figure 5.29: Loss in detection: plot of the detection before (the purple dots) and after (the purple crosses) step t . Dots closer enough to crosses are tracked to be the same person.

Component	Value
$p_{t-1,t-2}$	6
$p_{t,t-1}$	4
$\Delta p_{t,t-1}$	-2
$p_{common_{t,t-1}}$	3
$p_{new_{t,t-1}}$	1
$p_{lost_{t,t-1}}$	3
$\Delta R_{t,t-1}$	-2

Table 5.16: Loss in detection: analysis of the reward function components.

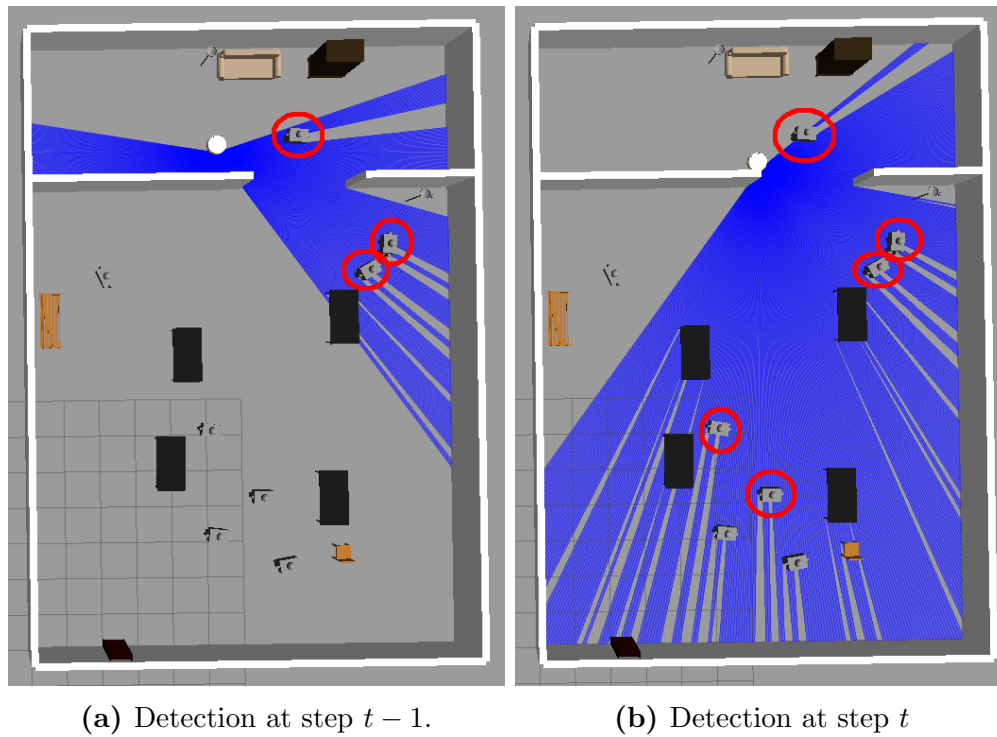


Figure 5.30: Gain in detection: on the left image and the right one the GAZEBO windows of the context before and after step t .

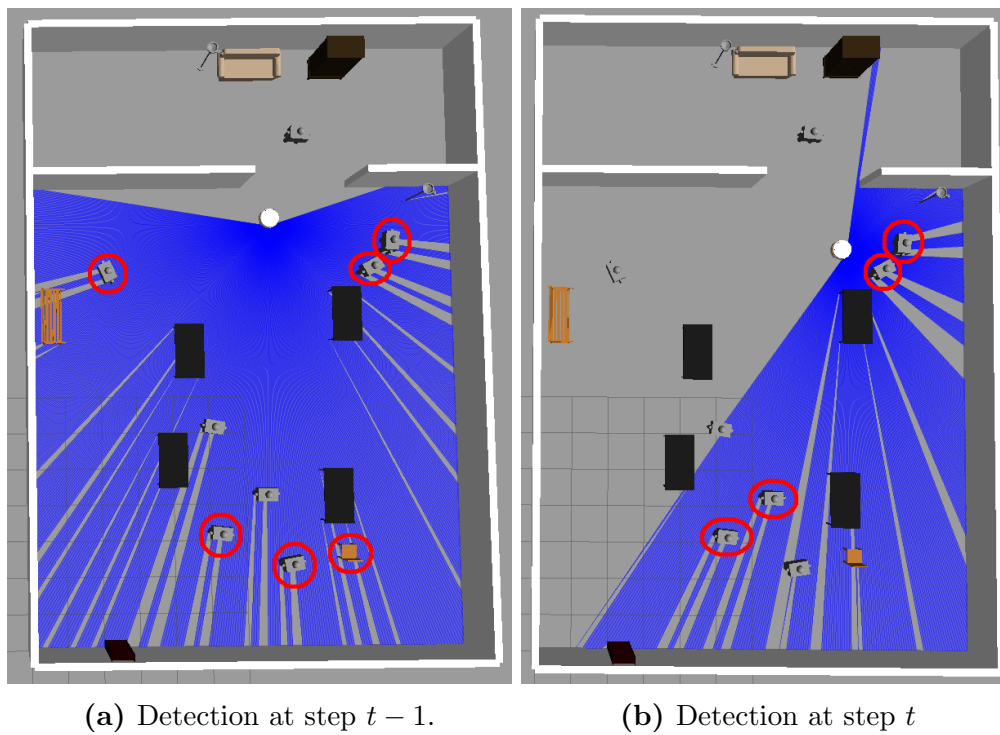


Figure 5.31: Loss in detection: on the left image and the right one the GAZEBO windows of the context before and after step t .

5.7 Learning from data

After the policy definition, it is interesting to capture correlations among data exploiting NN, to learn them. The result of NN training is a model that can drive an agent in an environment, not necessarily known, showing active perception behavior to maximize the number of people detected, trying to increase efficiency. Starting from all the basic elements of the policy, listed in the previous section, the structure of the network model and the parameters with training results are presented.

Since this method is an extension of the policy definition of the previous section, the main elements are the same. Running the policy for n_epochs and collecting main data for each action execution, then the network is trained as in Algorithm 16.

Algorithm 16 Learning from policy data runs

Input: Number of epochs n_epochs , number of steps n_steps

Output: Trained agent

```

1: for  $e \in n\_epochs$  do
2:   for  $t \in n\_steps$  do
3:     Load policy data run;
4:     Predict output;
5:     Compute loss;
6:     Backward update;
7:     Optimize network;
8:     Update the cumulative reward  $U_t$ ;
9:   end for
10: end for
    return Trained agent;

```

5.7.1 Neural Network (NN) for policy learning

The learning of this section trains a NN to provide in output a model to use. The input to the DQN is the representation of the current state of the environment in features. The network maps the input to the output, which is the set of values for the Poses¹³ data ($Position = [x, y, z]$, $Orientation = [x, y, z, w]$). Therefore, given the input state, the model must be able to predict the goal pose to reach to have the biggest increase in detection, according to the policy previously defined.

¹³For further details: https://docs.ros.org/en/noetic/api/geometry_msgs/html/msg/Pose.html .

The structure of the network (Figure 5.32) can be resumed as follows:

- The state has a shape (212×3) ;
- Given an input, a first forward pass, and a simple transposition makes it a vector $(1, 212)$.
- This is a feed-forward network in which the central part is made up of three groups of fully connected and Leaky Rectified Linear Unit (ReLU) activation function layers¹⁴;
- The last layer is fully connected, which action space is equal to seven since the goal poses the network has to predict are made up of seven elements.

5.7.2 Parameters for policy learning

The parameters that let the learning run are defined in Table 5.17.

n_inputs	$n_outputs$	$learning_rate$	n_epochs	$n_steps = n_data$
disc 212	7	0.0001	5000	1621 (DANGER)
212	7	0.0001	5000	1460 (DISCOVER)

Table 5.17: Parameters for the policy learning: the number of steps for a single epoch changes according to the number of data collected during simulated runs of the policy.

5.7.3 Results of learned policy

As can be seen in the figures, indices compared to evaluate the results of the learning of both the *DANGER* and the *DISCOVER* scenario are the loss values (Equation 5.26), the RMSE (Equation 5.27), and the MAE (5.28). Also, training time is added to the analysis.

$$\text{Loss MSE} = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad (5.26)$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2} \quad (5.27)$$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i| \quad (5.28)$$

¹⁴Here Leaky ReLU is used to allow negative data. The use of ReLU would have not led to negative numbers as a prediction of the network.

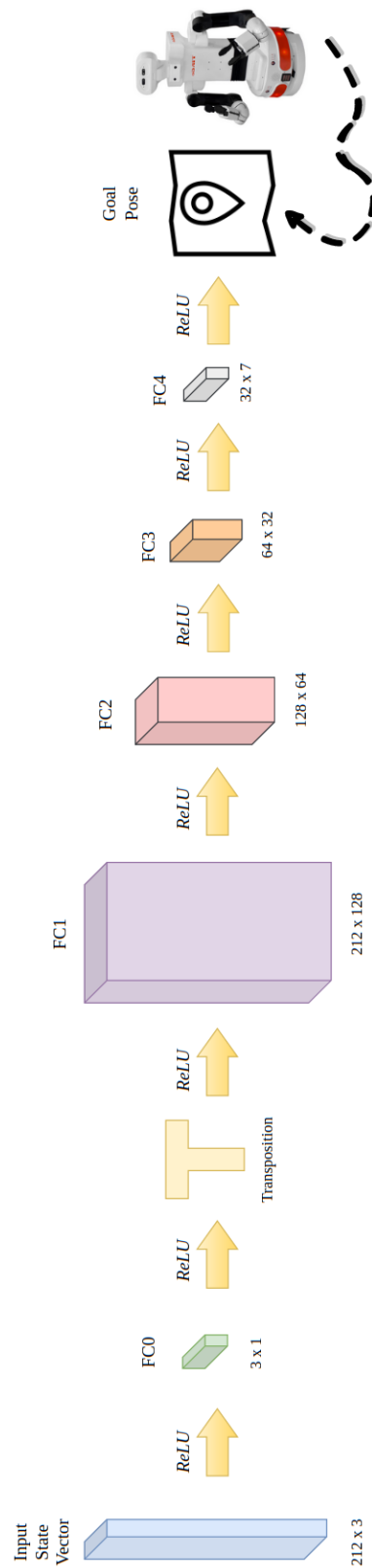


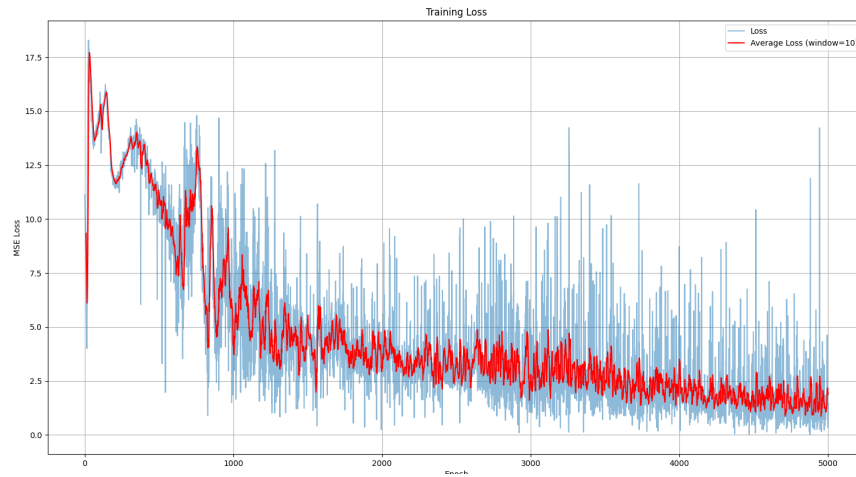
Figure 5.32: Structure of the NN network used to learn the policy.

<i>DANGER</i>	<i>MIN</i>	<i>MAX</i>	<i>AVG</i>
LOSS	0.0191	18.2871	4.6354
RMSE	1.3121	5.6898	1.8016
MAE	0.3180	3.0559	0.6631

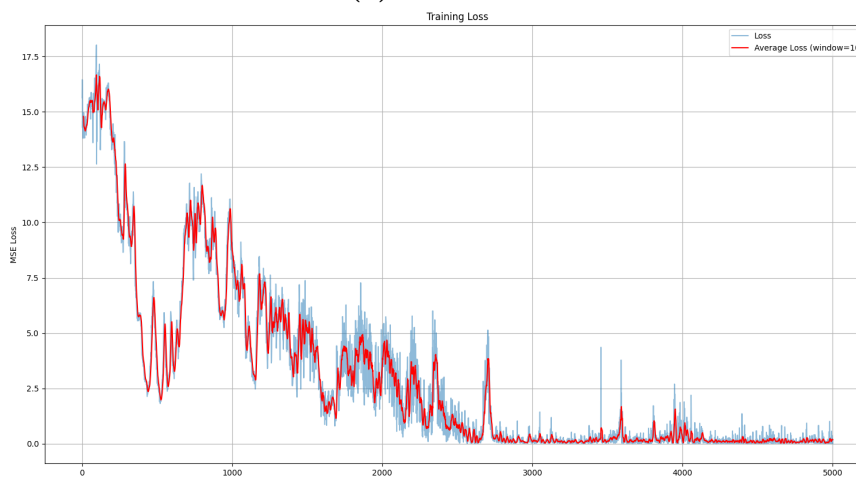
Table 5.18: Parameters for the policy learning (*DANGER* scenario): 33.00 minutes, 58.61 seconds.

<i>DISCOVER</i>	<i>MIN</i>	<i>MAX</i>	<i>AVG</i>
LOSS	0.0023	18.0214	3.0110
RMSE	0.6615	4.8578	1.3811
MAE	0.2558	2.9781	0.5892

Table 5.19: Parameters for the policy learning (*DISCOVER* scenario): 29.00 minutes, 31.47 seconds.



(a) *DANGER*.



(b) *DISCOVER*.

Figure 5.33: Scenario comparison: learning loss. In blue the loss is shown, while in red the average loss with a data window of 10 values.

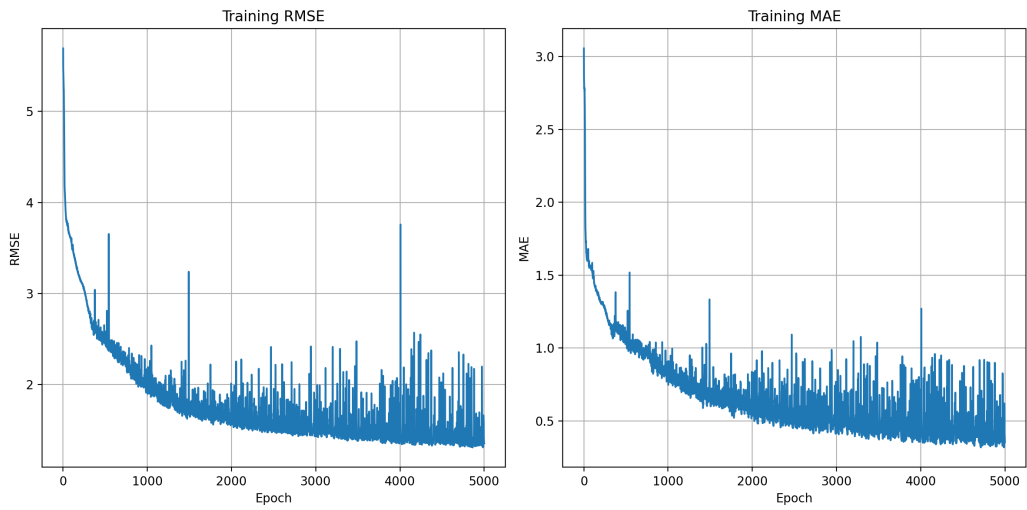


Figure 5.34: Scenario *DANGER*: RMSE and MAE indices.

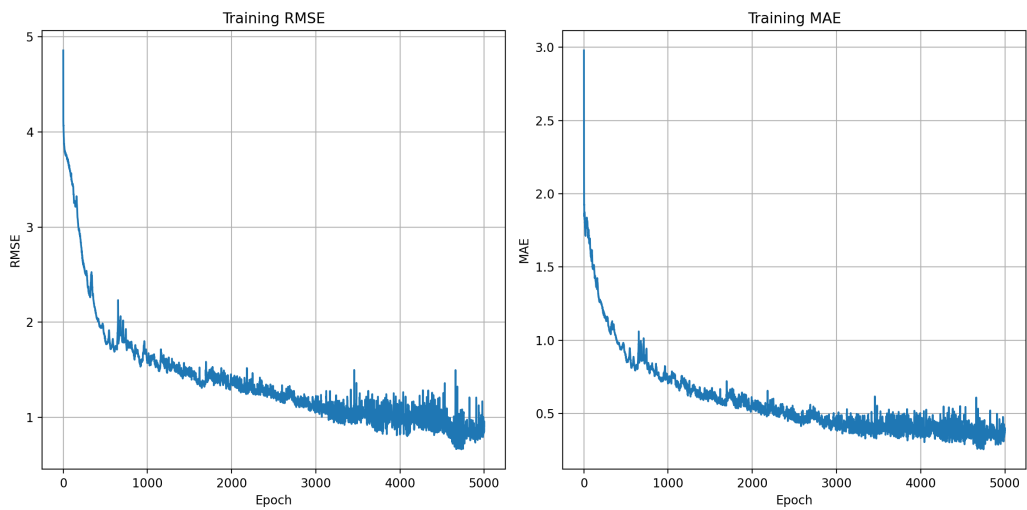


Figure 5.35: Scenario *DISCOVER*: RMSE and MAE indices.

5.8 RL Training

The RL Training method is explained here. As a result of this training, a model should be returned such that an agent that uses the model can navigate in an environment, not necessarily known, showing active perception behavior to maximize the number of people detected. With the term navigation, it is intended that the agent should be able to predict one of the four possible movement actions (forward, backward, turn left, turn right) to execute and accomplish the task.

The computations are done using the template code available in the *openai_ros* package (Section 3.1.5) and other Python modules (gym¹⁵, torch¹⁶, etc).

5.8.1 RL training elements

For a better comparison, here the main elements of the RL training for the movement actions prediction are listed, similar to what is in Section 5.6.

- *Scenarios*: the two types of scenarios are the same for consistency, thus also the weights associated with them;
- *State*: the current state s_i for step i is represented by the same vector as before;
- *Actions*: $a = \{\text{FORWARD, TURN LEFT, TURN RIGHT, BACKWARD}\}$. In the code the set of possible actions will be codified as $a = \{0, 1, 2, 3\}$;
- *Actions Priorities*: no action priorities are defined since the training is guided by the reward associated with the actions;
- *Reward function*: at each step t , the reward must be updated as in Equation 5.10 by adding to the current reward the one computed for the specific step (R_{sap}). Differently from the reward defined in Section 5.6.4, in this task, the robot has also to learn how to navigate. This is the reason why the current reward computed for the social active task R_{sap} , also a reward $R_{navigation}$ is added as a second part of the total current step reward;
- *Done conditions*: each step can end either as soon as the last step has been completed or when the DONE condition is achieved. This condition is a concatenation of three possible situations:

¹⁵For further details: <https://www.gymnasium.dev/index.html> .

¹⁶For further details: <https://pypi.org/project/torch/> .

1. The maximum number of episode steps allowed is reached;
 2. The robot reaches the maximum episode reward;
 3. The robot has detected the maximum number of people;
 4. The robot has crashed into some obstacle.
- *Clustering method*: no clustering algorithm is used, the agent must learn the right way to understand whether to choose an action in place of another, using the information of detection to compute the reward;
 - *Group utility function*: no group utility function is defined to let the agent free to learn its policy;
 - *ID assignment*: another skipped part is the assignment of IDs to points;
 - *Memory list*: the agent learns alone whether to use a similar mechanism or not;
 - *Working policy*: given a current context (step), the robot must follow the high-level steps of Algorithm 17.

Algorithm 17 Robot RL training step

Input: LiDAR sensor data at time t

Output: Current iteration reward R_{t+1}

- 1:
 - 2: Compute sensor data classification and people detection (Section 4.3);
 - 3: Save the detections and points labeled as undefined;
 - 4: Build the state vector;
 - 5: Choose the action using Q-learning;
 - 6: Execute the action;
 - 7: Do again detection;
 - 8: Compute the current iteration reward;
 - 9: Train the DQN;
 - 10: Check the DONE conditions;
 - 11:
- return** Reward R_{t+1} ;
-

5.8.2 Deep Q-Network (DQN) for RL training

The RL training of this section runs to train a DQN to provide as output a model to use. The input to the DQN is a representation of the current state of the

environment in features. The network maps the input to the output, which is a set of action-values (Q-values) associated with each possible action that the agent can take in the given state. These Q-values indicate the expected cumulative rewards the agent can obtain by taking each specific action in that state: at step i , the network has to learn the function $f : a_i = f(s_i)$ where s_i is the current state and a_i is the action to take to reach state s_{i+1} from state s_i .

The structure of the network (Figure 5.36) can be resumed as follows:

- The state has a shape (212×3) and at each training step the network has been trained using batches of 16 elements (e.g.: in Figure, the network has been shaped to accept input batches of $n = 4$ states);
- Given a batch of size n , the input shape is $(n, 212, 3)$. Before passing forward the input state batch, it is subject to a reshape through the first fully connected layer, so that the current shape is $(n, 212, 1)$. All the n batches are concatenated one on the right of the other (concatenation at dimension 1) so that their shape is equivalent to $(212, n)$. Then, a simple transposition makes them to be matrices $(n, 212)$.
- This is a feed-forward network in which the central part is made up of three groups of fully connected, 1D batch normalization¹⁷ and ReLU activation function layers;
- The last layer is fully connected, which action space is equal to four since there are four possible movement actions the network can predict.

5.8.3 RL training parameters

This subsection gathers all the parameters that have been set for the training and used from the beginning. The parameters tuned during learning are in the next section (5.8.4). Other parameters like reward weights matrices for SAP have not been repeated.

All these tables are built respectively over the parameters used for:

- Training of the DQN (Table 5.20);

¹⁷This type of layer improves the training and convergence of deep neural networks through the normalization of the input across the batch dimension. Learning is stabilized and accelerated.

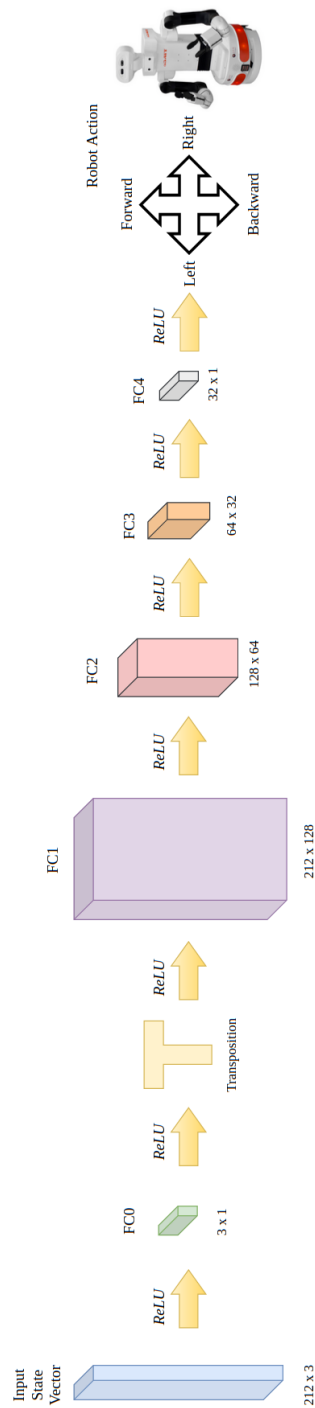


Figure 5.36: Structure of the DQN network proposed for RL training.

- RL training (Table 5.21);
- Task environment definition (Table 5.22);
- Social Active Perception (Table 5.23).

γ	n_{steps}	$batch_{size}$	lr	ϵ_{start}	ϵ_{end}	ϵ_{decay}
0.999	150k	$bs = 16$	1e-3	1.0	0.1	80k

Table 5.20: Parameters for the DQN network training.

α	γ	ϵ	$\epsilon_{discount}$	$n_{episodes}$	n_{steps}	$step_{running_time}$
0.1	1.0	0.9	0.999	500	1000	0.06s

Table 5.21: Parameters for the Q-learning and the RL training.

$n_{actions}$	$n_{navigation}$	$state_{shape}$	$R_{forward}$	R_{turnL}	R_{turnR}	$R_{backward}$	$R_{end_episode}$	$speed_{forward}$	$speed_{turn}$
4	6	$(bs, 212, 3)$	+5	+1	+1	-5	-200	$[3.5m/s, 0rad/s]$	$[0.5m/s, 1.0rad/s]$

Table 5.22: Parameters for the task environment definition.

$distance_{robot_group}$	$distance_{random_step}$	$distance_{clustering}$	(μ, σ)	avg_speed_{human}
3.6m	1.0	2.4	$(0.06663865, 0.05414631)$	0.12m/s

Table 5.23: Parameters for the Social Active Perception task.

5.8.4 Results of RL training

The approach based on RL training has been executed for the DISCOVER scenario for 500 steps and 1000 steps. Figure 5.37 and Figure 5.38 show that we have not achieved satisfying results. It means that the model has not learned well enough even though the training ran for two days. Of course, more training time is required.

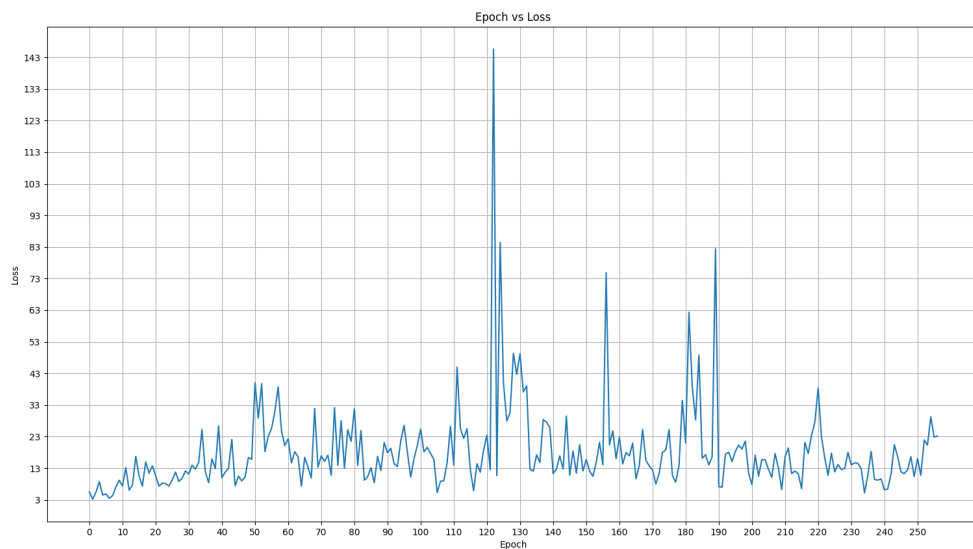


Figure 5.37: Loss achieved during RL training.

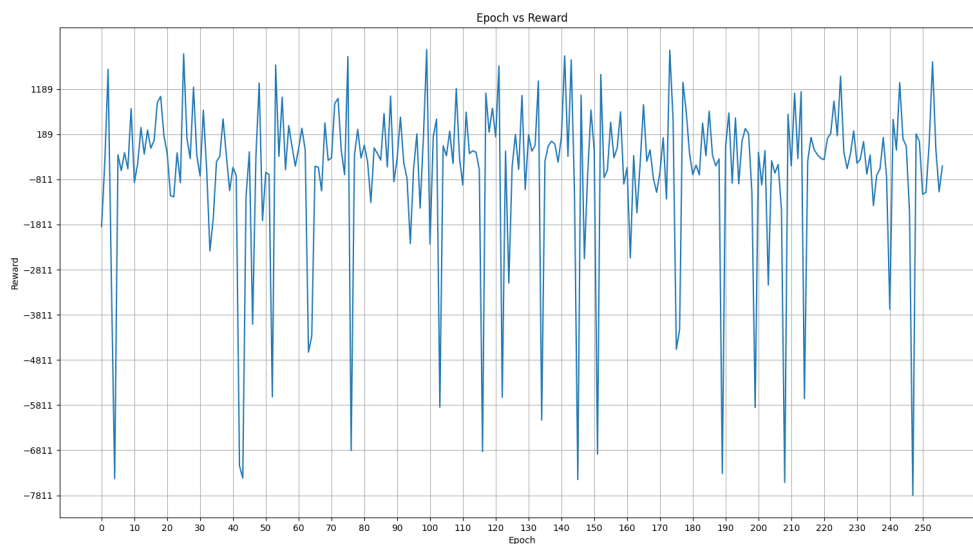


Figure 5.38: Reward achieved during RL training.

5.9 Simulated World

To run the policy and the RL training, a world for the simulations to let the robot learn is needed. The world must be compliant with the use it has to be exploited for. Indeed, the design and the use of this world are mainly focused on the training phase, without putting too much effort into details. The features to put in the context are the following:

- Large environment: a big location that provides the classical services present in these places;
- Groups of people: they can be families or friends that are going to any shop, or the info points, etc;
- Queues of people: for example, they are waiting for the elevator to change the floor;
- People waiting: people standing next to the chairs;
- People walking: people moving around, simulated with the help of PedSim simulator (Section 3.1.4) that are better displayed in Figure 5.40;
- People who are doing a specific activity: for example, they are buying something from the vending machines are either speaking in groups or waiting at the info point operator, etc;
- Alone people: these can be modeled as groups of people with size 1;
- Private room: the classical room, not accessible by clients, where the employees rest or put in order work tools.

In Figure 5.39 the chosen world is displayed. It is the context of an airport terminal, which is a vital part of an airport where passengers check in, pass through security, wait for their flights at departure gates, and retrieve luggage upon arrival. This custom world presents all the features listed above.

As previously mentioned in Section 4.5, the humans are simulated with a custom simple model to increase detection performances as much as possible.

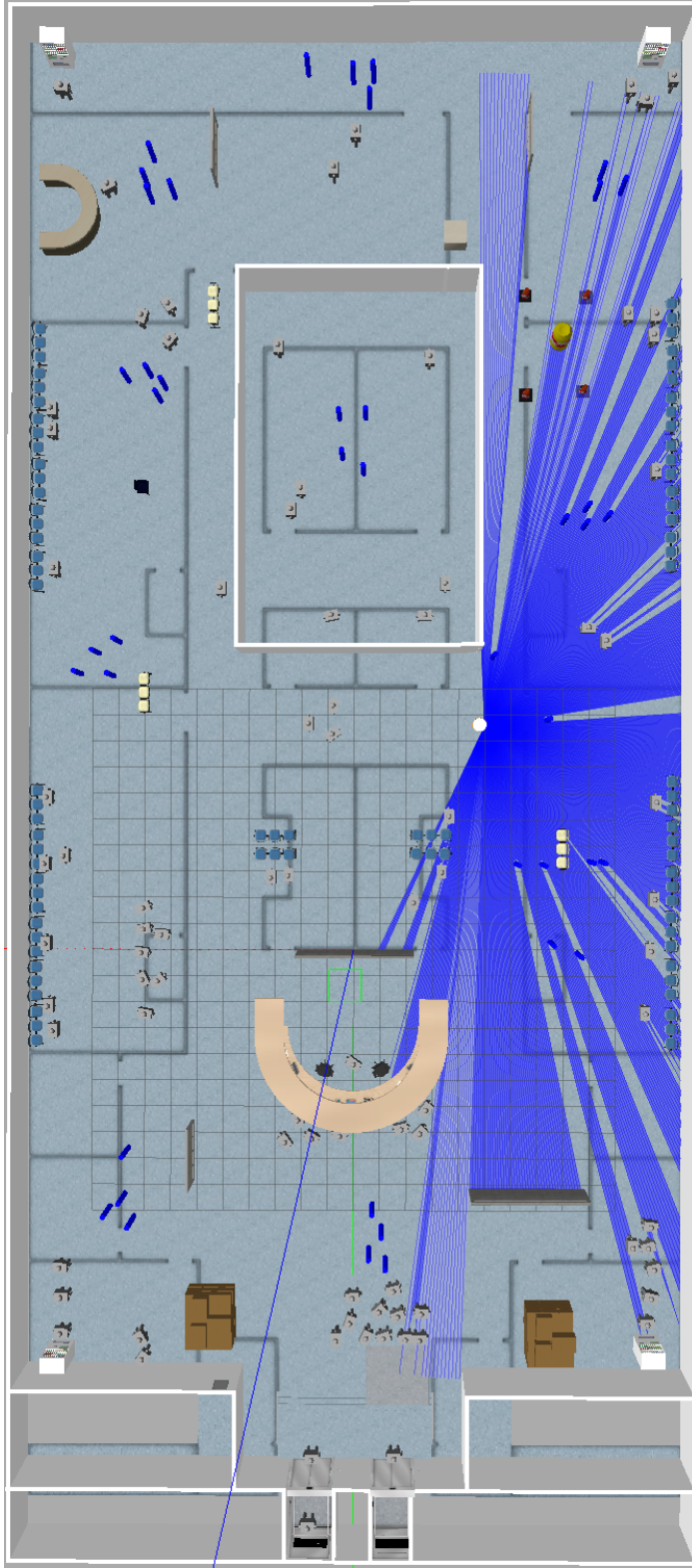


Figure 5.39: GAZEBO view of the simulated world: an airport terminal. Blue cylinders represent the agents that move around the environment. In this world there are more than forty agents to ensure high dynamism. The other elements are obstacles, walls, furniture, and static people.

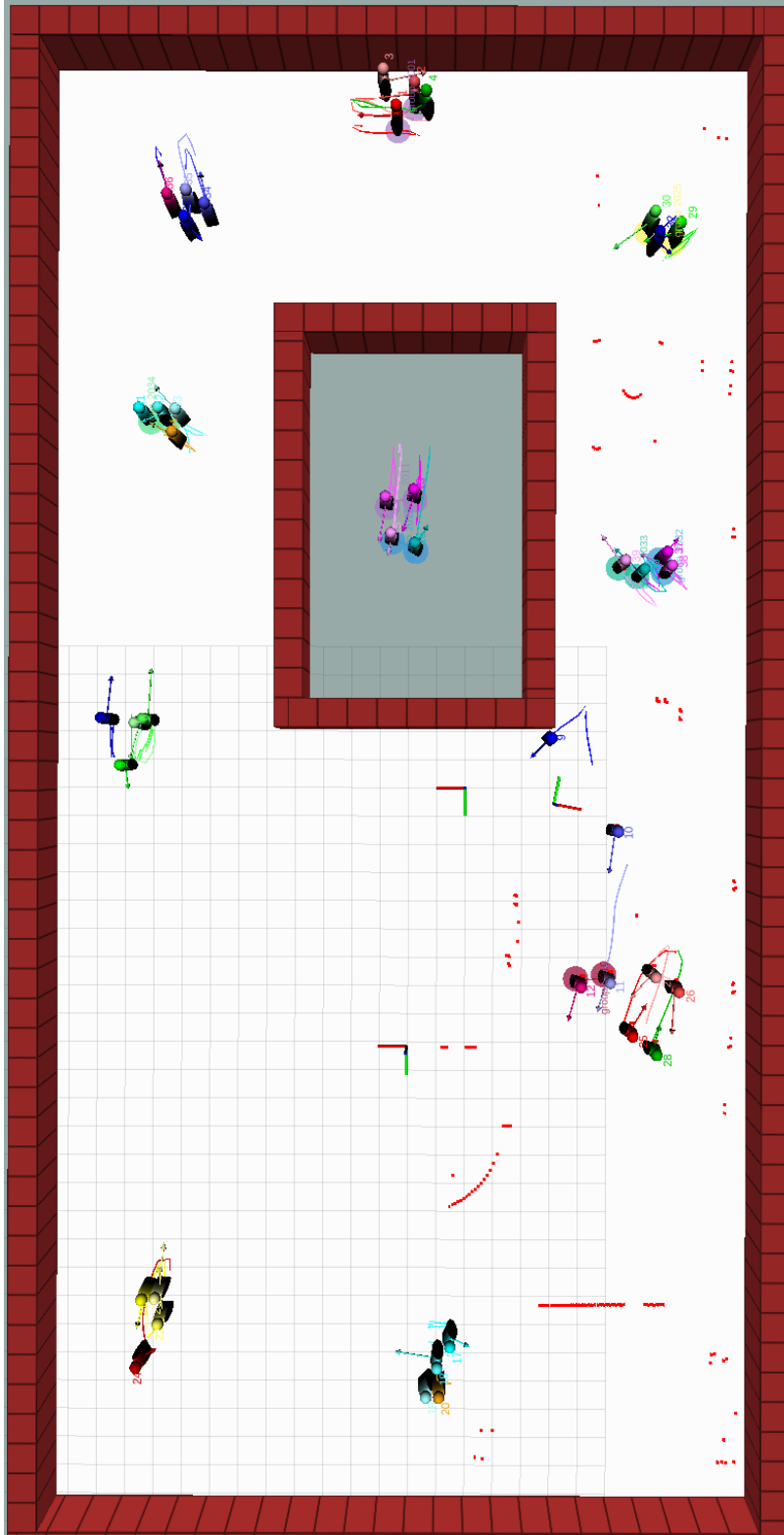


Figure 5.40: RViz view of the pedestrians in the simulated world. Since PedSim works with SFM, main obstacles such as walls are properly defined. The white ground is the area in which the robot can move around. The top-left reference frame represents the world frame, the middle-right frame refers to the map frame, and the bottom-right frame represents the base_laser_link. Red dots are the laser points that detect obstacles. Laser dots that belong to wall detection are hidden by PedSim red walls, but they are properly handled.

In this work the main features are the high number of obstacles, contexts and the dynamism. All dynamic agents are spawned in the world by PedSim, which allows social navigation. Indeed, in this simulated world the agents can avoid collisions during navigation both between them, between them and the robot, and against walls. The planned path in case of possible human collision is modified a little bit, while for human-robot avoidance the paths change more. Instead, since walls are enlarged to ensure secure human navigation, agents can avoid also environmental boundaries (Figure 5.41).

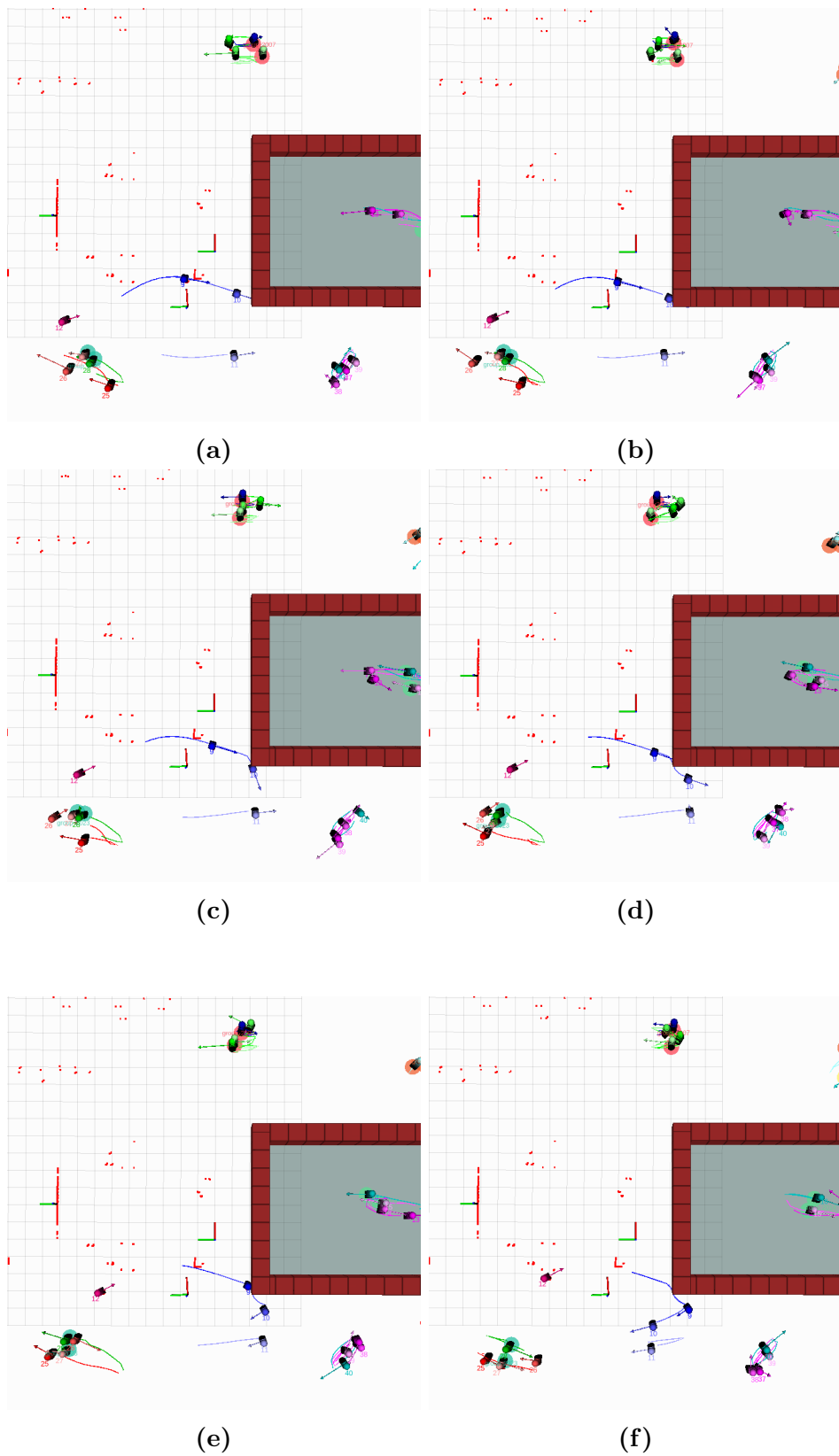


Figure 5.41: From (a) to (f): the path planned of PedSim agents with ID 9, 10, 11, and 12 are modified such that to observe social behaviors in navigation. For agents 9 and 10 this behavior is much more highlighted since they are closer to the robot than agents 11, 12.

Chapter 6

Experimental Results

This chapter collects all the tests performed on the real robot and the related experimental results. These tests have been chosen to validate whether the work contributions described in Chapter 5 allow the robot to behave in an effective way thanks to the active perception. Precisely, all the tests have been carried out with the approach of the deterministic policy described in Section 5.4.1.

As already anticipated, the chosen robot is a TIAGo (in particular, a TIAGo++) located in the Autonomous Robotics Laboratory of the University of Padova.

The testing phase is structured as follows:

- All tests have been performed for both scenarios *DISCOVER* and *DANGER* in a real environment (see Section 6.1);
- All tests have been repeated at least ten times, except for the one with only obstacles (in this case one run has been considered sufficient since it is out of the scope of this thesis);
- Each single trial has been stopped after the robot has detected all people¹ or when the robot delocalized²;
- Ten trials of a single test have been analyzed to extract the results reported in the next sections.

¹All people present in the particular test have been detected by the robot during the actions.

²During the navigation, the robot is no longer able to map correctly its pose in the working environment. This leads to a wrong association between robot pose and data measured by the robot through its sensors (e.g.: odometry from navigation sensors, laser points about the environment from LiDAR sensor, ...).

6.1 Test Environment

The map of the whole test environment is represented in Figure 6.1. Since the focus of this thesis is on crowded environments, the test environment must simulate a crowded world. Hence, a small area has been chosen to place people and let the robot perceive and act given the difficulty in recruiting a big number of people. For this reason, all tests have been performed only in the area inside the red box, and all the next figures will take into account only what happens inside that region.



Figure 6.1: Map of the whole test environment. The map frame is located in the origin. The area used for the tests is the one inside the red box: width 7.85m and height 2.55m.

The obstacles used in the tests have been collected to appear as in Figure 6.2.



Figure 6.2: Obstacles used in the environment: (a) Bin; (b) Small table, pole, and two cones. All these obstacles were used at the same time only once. Notice the white panels to hide desks, chairs, and other objects that are not part of the experiments.

6.2 Test Configurations

In this section, all six types of experiments are described and analyzed. These specific configurations have been tested to highlight or stress a peculiar behavior of the robot.

In detail, the following points list the six configurations (Figure 6.3):

1. *Only obstacles*: the robot has to move around the test area to verify which obstacles are robust to the method or often lead to false positives. For this simple test, it is considered only one run;
2. *Group vs. single target*: in the scene, there are three people where two form a group and the third one is alone;
3. *Queue of targets*: three people are standing in a queue, and the person nearest to the robot occludes one leg each of the two people behind him in the queue. The remaining legs must help the robot detect all people;
4. *Occlusion to target*: a small desk occludes the right leg of the standing person. The visible leg has to suggest to the robot to get closer and detect successfully the person;
5. *Occlusion to object*: similar to point (4), but this time a pole next to the small desk has to try to confuse the robot about false positive detection; while getting closer the robot has to get more confident about the pole as a true negative;
6. *Double target occlusion with objects*: points (4) and (5) are mixed to stress the robot about uncertainty on the standing person that has a leg completely occluded by the small desk and the other one partially occluded by the pole.

The behaviors described above represent the expected behaviors of the robot, but the specific scenario in which it is deployed to work has to change it accordingly:

- *DISCOVER*: the robot is allowed to move around the environment and discover as much information as possible, even though it has to find people;
- *DANGER*: the robot is strongly expected to have a more direct movement towards people detected, before moving around the environment to look for people.

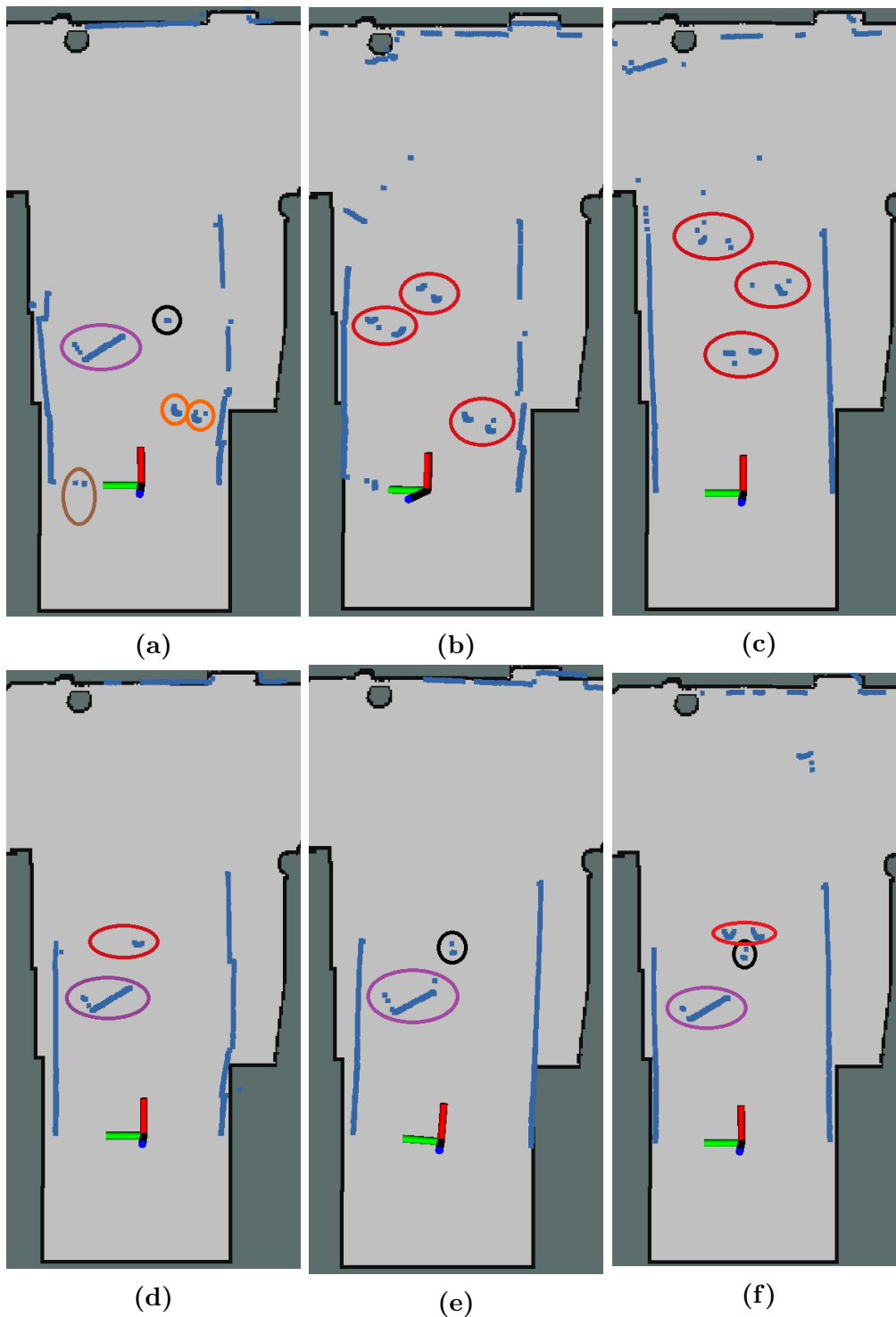


Figure 6.3: RViz visualization of the six types of the setup for the experiments: (a) only obstacles; (b) group vs. single target; (c) queue of targets; (d) occlusion to target; (e) occlusion to object; (f) double target occlusion with objects. Only the used part of the map environment is shown. Laser scan points are the blue dots. The reference frame represents the starting pose of the robot. Colors: red for a person, orange for a cone, purple for the small desk, black for the pole, and brown for the bin.

The parameters for the people detection template have been set experimentally, starting from those computed in simulations (see section 4.2). Hence, the final reference parameters mean and standard deviation are:

- μ_r : 0.0666;
- σ_r : 0.0541.

6.3 Evaluation metrics

To evaluate the tests performed for this thesis, the following evaluation metrics for active perception were computed:

- *People detected*: maximum number of people found per run, in both scenarios;
- *Accuracy*: formulated as in Equation 6.1, the accuracy $A(test)$ has been measured on a percentage scale as the accuracy achieved both in all tests and in the whole scenario;

$$A(test) = \begin{cases} \frac{1}{10 \cdot n(test)} \cdot \sum_{i=1}^{n(test)} a(test, p_i), & \text{if } true_people(test) > 0 \\ 1 - \left(\frac{1}{10} \cdot \sum_{i=1}^{n(test)} a(test, p_i) \right), & \text{if } true_people(test) = 0 \end{cases} \quad (6.1)$$

$$a(test, p_i) = \begin{cases} 1, & \text{if person } p_i \text{ has been found during the } test \\ 0, & \text{if otherwise} \end{cases} \quad (6.2)$$

where $n(test)$ is the number of true people present in the test, $a(test)$ is the binary value associated with a person if detected at least one time during the test, as in Equation 6.2. Note that in case no people are involved in the test, then the accuracy is computed through the error about false positive detection;

- *Execution Time*: comparison of duration per run and average duration per run, both scenarios;

- *Actions executed*: minimum number of actions required to detect the maximum number of people found and the total number of actions done before the end of the run;
- *Qualitative evaluation of active perception*: in terms of robot-people distance and time of the test, this evaluation has to verify the increase in confidence about people detection when the robot gets closer to people, and the time required by the system to verify the presence of people. More precisely, this type of evaluation has to show how the regions of undefined points become likely as soon as the robot gets closer to those areas, during the test, according to the scenario. Finally, if there are people, the robot has to detect them with a higher confidence value, otherwise, points have to be discarded in the next robot's motion. The qualitative analysis has to support the hypothesis that active perception is suitable for the robot to convince about the state of the environment in a given scenario. Moreover, the analysis has to verify the evidence of the hypothesis in real contexts;
- *Correlation over data*: data about people-robot distances and confidences in people detection of all runs and all tests in both scenarios have been collected. These data have been analyzed to verify their linear correlation since we hypothesize the confidence of detection has to increase (and the uncertainty has to decrease) when the distance decreases (and the robot gets closer to people or objects). These correlation has been computed using Model-Implied Temporal Associations³.

6.4 Results

In this section the test results are collected and organized in subsections. Each result compares the performances achieved in a particular test, for both scenarios. Then, two sections about general comparison of the two scenarios end the chapter.

6.4.1 Results on only obstacles

First of all, a test about how the robot perceives the environment is required to understand where possible sources of false positive can be located.

³Model-implied temporal correlations and covariances measure the self and cross-lag associations between measurement and state variables in a state-space model.

Figure 6.4 shows the RViz window in which the robot is located in three different poses. The only obstacles that the robot detects very often as a false positive are the two cones, while our approach is very robust to the others. Since this test focuses only on the evaluation of false positive detection, no difference according to the scenario is considered.

For the image of the real world please refer to Figure 6.2.

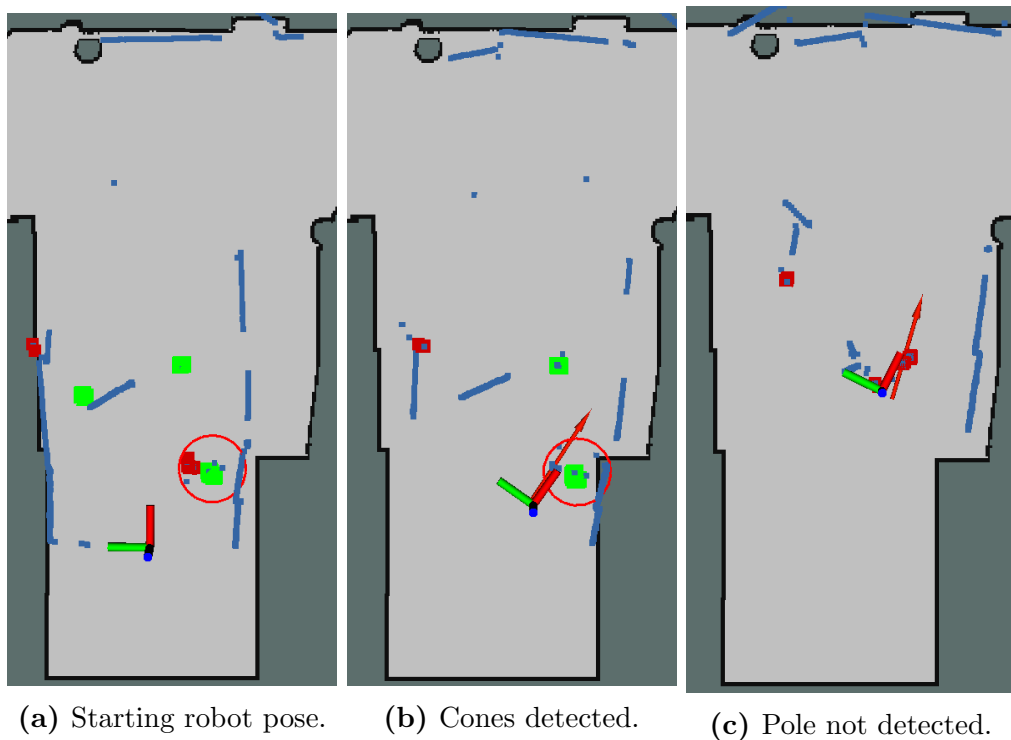


Figure 6.4: RViz representation of the three poses reached by the robot. Only the experimental area is shown. Laser scan points are the blue dots. The reference frame represents the current pose of the robot. Colors: red for points labeled as UNLIKELY, green for points labeled as LIKELY, and a red circle indicates the position of a detected person by the hybrid system composed of DR-SPAAM + preprocessing.

6.4.2 Results on group vs. single target

From the results in the following figures, this type of test confirms that in the DANGER scenario the robot pays more attention to people than in the other scenario. Indeed, in the left column of all figures detection are very good (Figures 6.5, Figure 6.6), and people are detected from the beginning (Figure 6.7, Figure 6.8). This leads to a smaller number of actions to do, even though the mean durations result to be very similar (Figure 6.9).

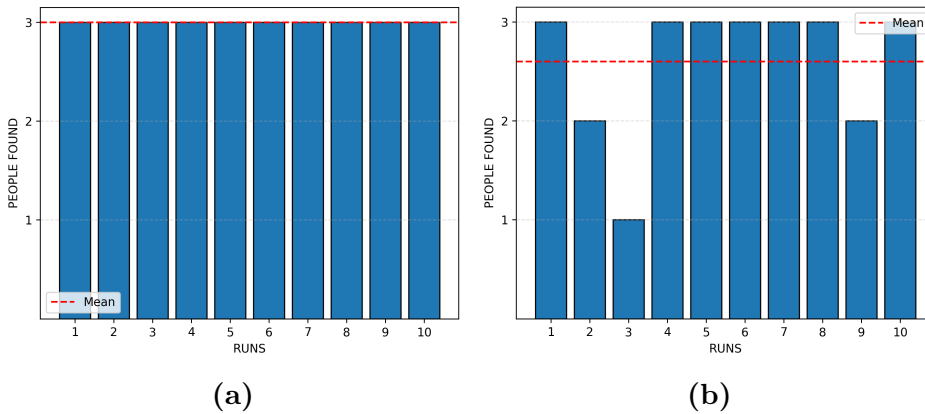


Figure 6.5: Comparison of people found in the ten runs. The red dotted line is placed in correspondence with the mean of the people found. Real number of people in the test is three. (a) Scenario DANGER; (b) Scenario DISCOVER.

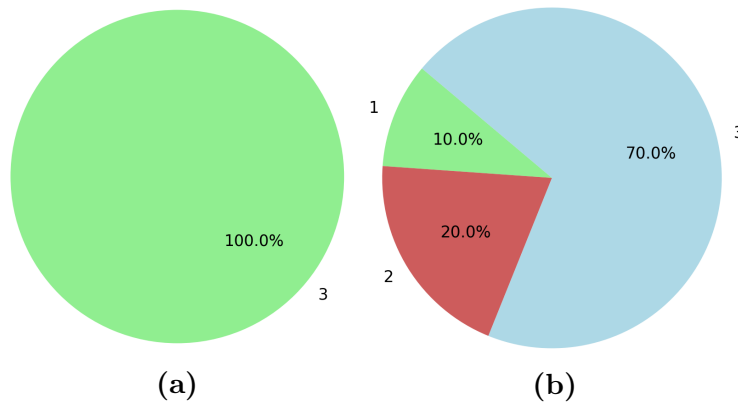


Figure 6.6: People found at the end of all runs. Classes correspond to the number of people detected. (a) Scenario DANGER; (b) Scenario DISCOVER.



Figure 6.7: Actions required to detect people found vs. total number of actions. The red dotted line is placed in correspondence with the mean of the actions required to detect people found, while the purple one is placed to represent the mean of the total number of actions executed. (a) Scenario DANGER; (b) Scenario DISCOVER.

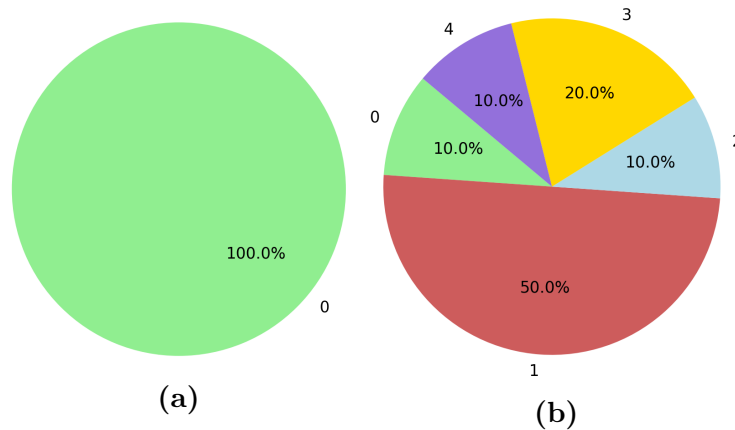


Figure 6.8: Minimum number of actions required to find the people found for all runs. Classes correspond to the number of people detected. (a) Scenario DANGER; (b) Scenario DISCOVER.

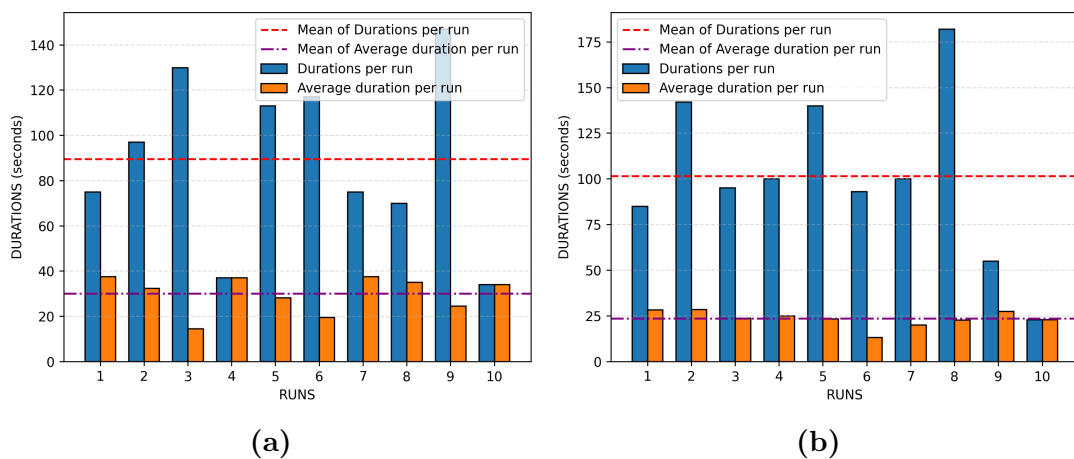


Figure 6.9: Durations per run vs. average duration per run. The red dotted line is placed in correspondence with the mean of the durations per run, while the purple one is placed to represent the mean of the average duration per run. (a) Scenario DANGER; (b) Scenario DISCOVER.

Figure 6.10 and Figure 6.11 are strictly related because they refer to the same test run in a given scenario. The first figure shows the evolution of people detection confidence over time and robot-people distance: black and empty circles have to become black and filled (since UNDEFINED points have to become LIKELY points), until some stars appear if the robot gets closer to people, otherwise they have to vanish. Moreover, the number of detection in the DANGER scenario is higher. This follows by the hypothesis in which, in that scenario, the robot has to pay more attention on people. Instead, the figure below shows the evolution of the path of the robot in blue, where the direction is indicated by the black arrow. People are still represented by stars.

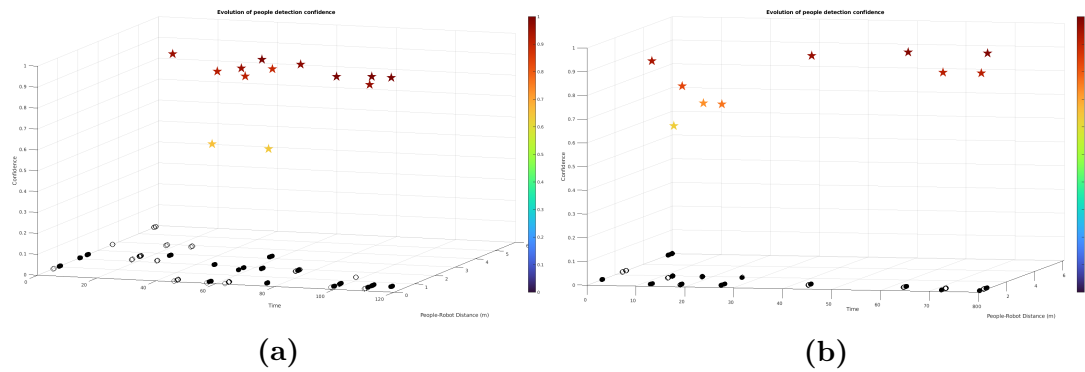


Figure 6.10: Test: group vs. single target. Comparison of the evolution of people detection confidence over time and robot-people distance. Black and empty circles are the points labeled as UNDEFINED. Black and filled circles are the LIKELY points. Stars represent people detected at a given time and robot-people distance. The color scale on the right represents the confidence level associated with a person detection. (a) Scenario DANGER; (b) Scenario DISCOVER.

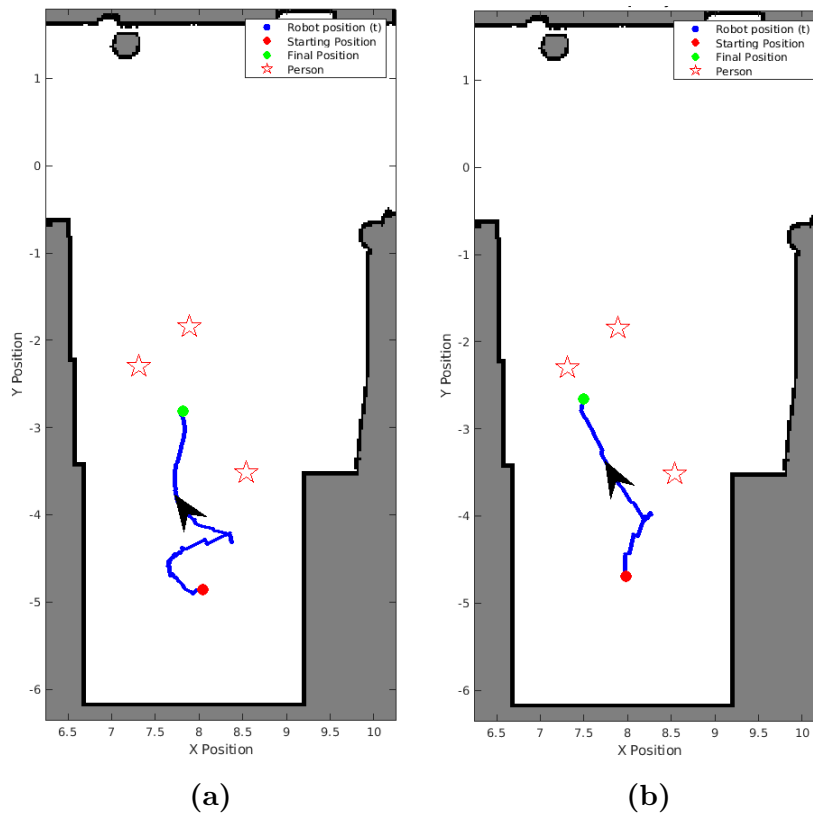


Figure 6.11: Test: group vs. single target. A blue trajectory is the path of the robot during a test run. The black arrows indicate the direction of motion. The red and green points are the starting and the final positions. Stars represent people. (a) Scenario DANGER; (b) Scenario DISCOVER.

6.4.3 Results on queue of targets

From the results in the following figures, this type of test confirms that in the DANGER scenario the robot is attracted more by the two people in the queue that it can detect better. Indeed, it is a bit more difficult to find new people (the third one it often misses) while working in the DANGER scenario (Figures 6.12, Figure 6.13). However, in the DANGER scenario, the robot detects people a little bit before than in the other scenario (Figure 6.14, Figure 6.15). The mean durations of actions are still very similar (Figure 6.16).

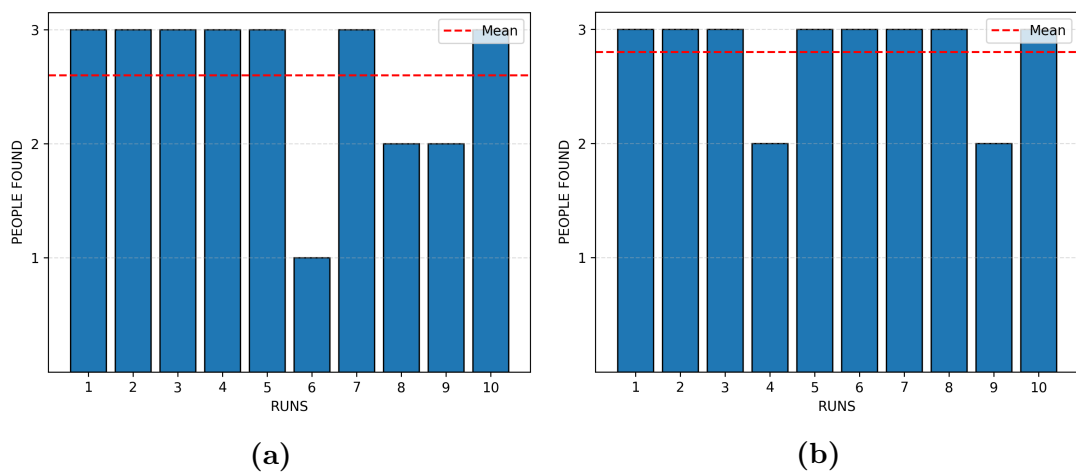


Figure 6.12: Comparison of people found in the ten runs. The red dotted line is placed in correspondence with the mean of the people found. Real number of people in the test is three. (a) Scenario DANGER; (b) Scenario DISCOVER.

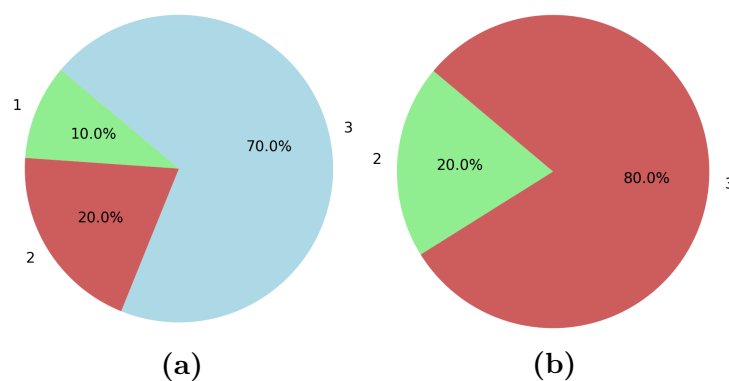


Figure 6.13: People found at the end of all runs. Classes correspond to the number of people detected. (a) Scenario DANGER; (b) Scenario DISCOVER.

Figure 6.18 and Figure 6.17 are strictly related because they refer to the same test run in a given scenario. The first figure shows the evolution of people detection confidence over time and robot-people distance: black and empty circles have to become black and filled (since UNDEFINED points have to become LIKELY

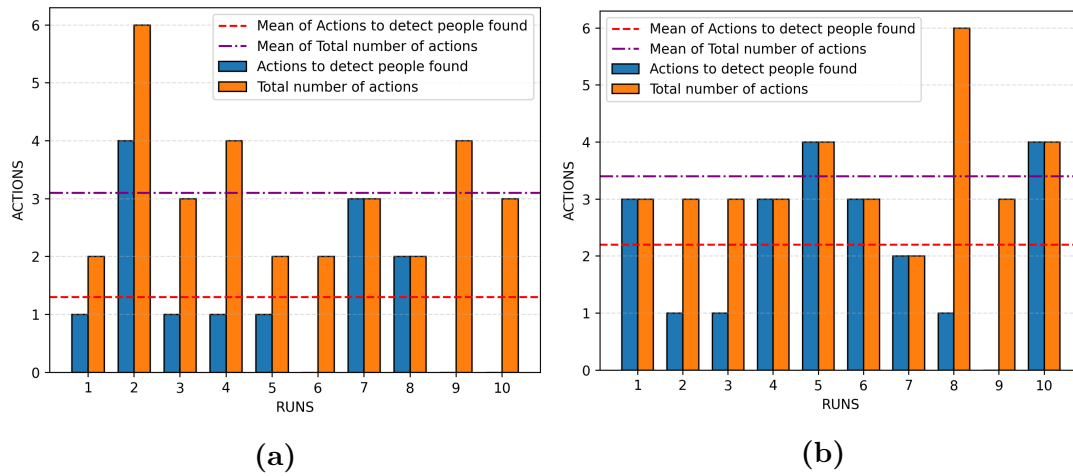


Figure 6.14: Actions required to detect people found vs. total number of actions. The red dotted line is placed in correspondence with the mean of the actions required to detect people found, while the purple one is placed to represent the mean of the total number of actions executed. (a) Scenario DANGER; (b) Scenario DISCOVER.

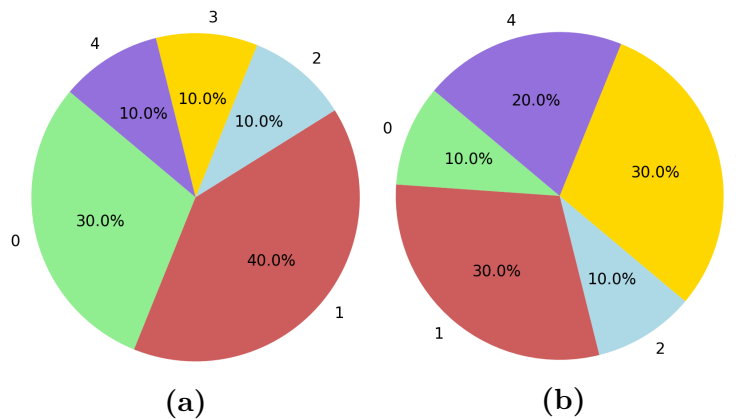


Figure 6.15: Minimum number of actions required to find the people found for all runs. Classes correspond to the number of people detected. (a) Scenario DANGER; (b) Scenario DISCOVER.

points) until some stars (i.e., people identified also by the detector) appear if the robot gets closer to people, otherwise they have to vanish. Moreover, the detection in the DANGER scenario have higher confidence. It means the behavior of the robot has been successful in accomplishing the task of people detection, keeping a good level of confidence. This follows the hypothesis in which, in that scenario, the robot has to pay more attention to people. Instead, in the other one, detection have more variable levels of confidence, as allowed for the discovery behavior. In addition, the figure below shows the evolution of the path of the robot in blue, where the direction is indicated by the black arrow. People are still represented by stars. The motion of the robot in the top-left of Figure 6.17(a) where it approaches the person is very different from the one in the DISCOVER

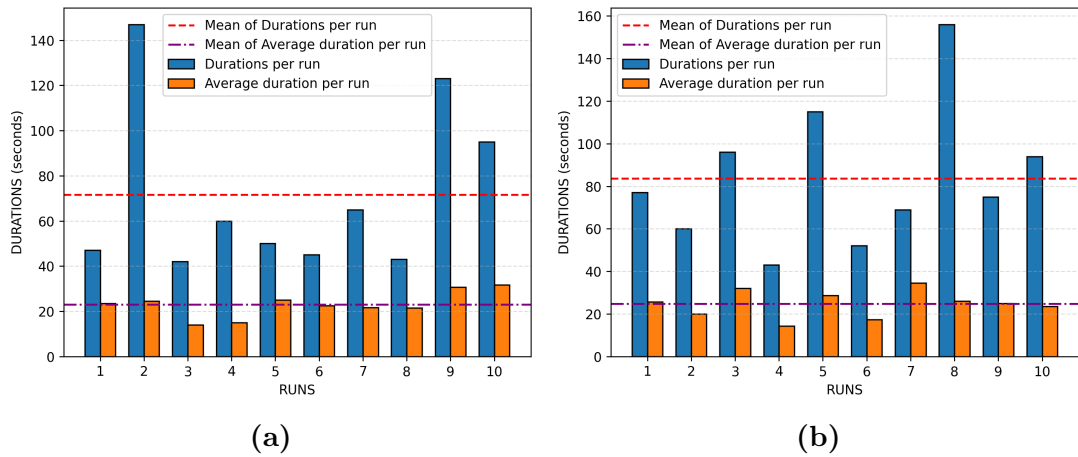


Figure 6.16: Durations per run vs. average duration per run. The red dotted line is placed in correspondence with the mean of the durations per run, while the purple one is placed to represent the mean of the average duration per run. (a) Scenario DANGER; (b) Scenario DISCOVER.

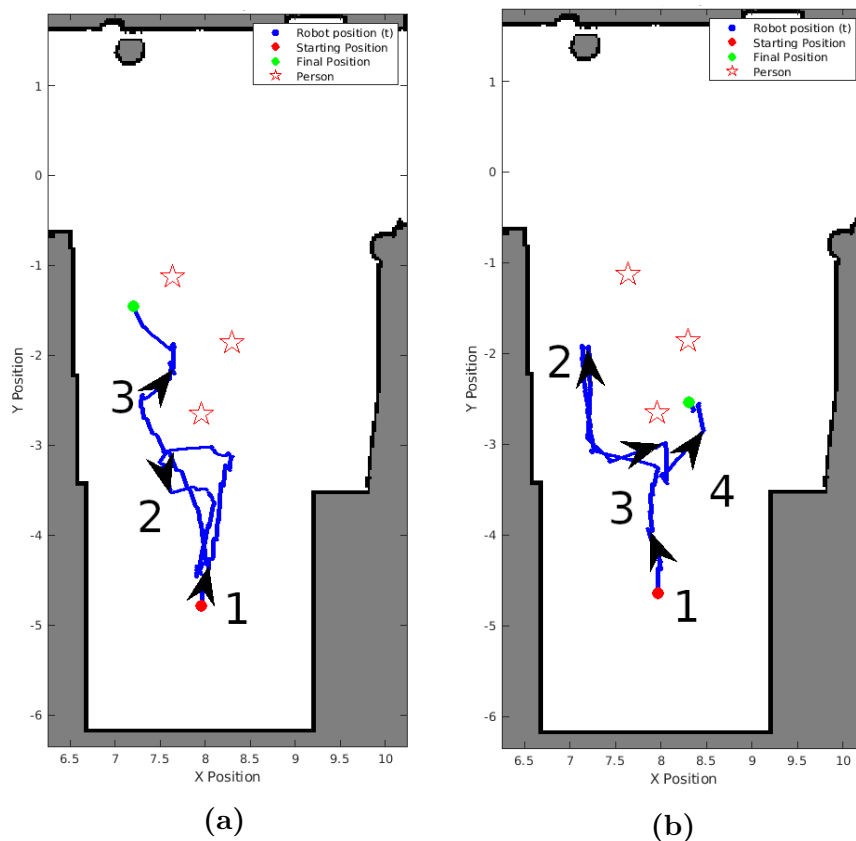


Figure 6.17: Test: queue of targets. A blue trajectory is the path of the robot during a test run. The black arrows indicate the direction of motion. The red and green points are the starting and the final positions. Stars represent people. (a) Scenario DANGER; (b) Scenario DISCOVER.

scenario, where the robot turns and goes to discover the person in the right-center of Figure 6.17(b).

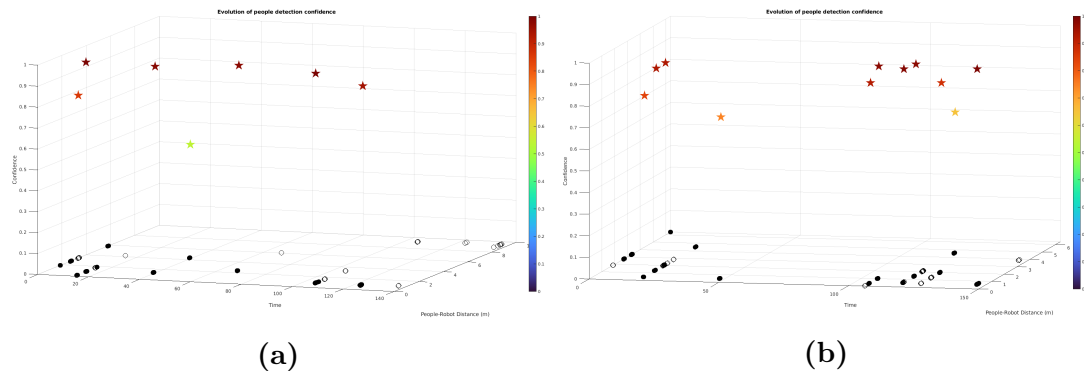


Figure 6.18: Test: queue of targets. Comparison of the evolution of people detection confidence over time and robot-people distance. Black and empty circles are the points labeled as UNDEFINED. Black and filled circles are the LIKELY points. Stars represent people detected at a given time and robot-people distance. The color scale on the right represents the confidence level associated with a person detection. (a) Scenario DANGER; (b) Scenario DISCOVER.

6.4.4 Results on occlusion to target

The results in the following figures are quite similar, it does not matter the scenario. The occluded person is always detected very soon (Figures 6.19, Figure 6.20). In the DISCOVER scenario, some different actions are performed, but it is good because of the working principle (Figure 6.21, Figure 6.22). This is also reflected in the mean durations of actions (Figure 6.23).

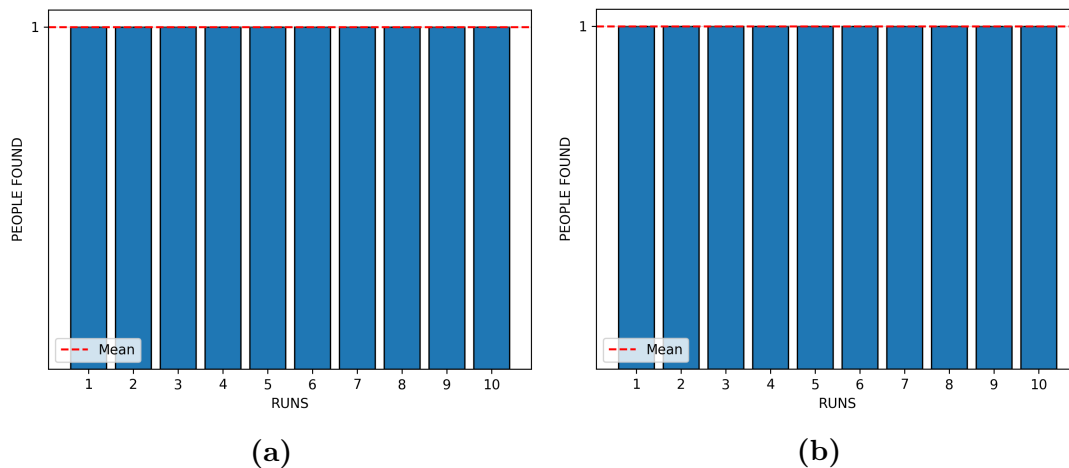


Figure 6.19: Comparison of people found in the ten runs. The red dotted line is placed in correspondence with the mean of the people found. Real number of people in the test is one. (a) Scenario DANGER; (b) Scenario DISCOVER.

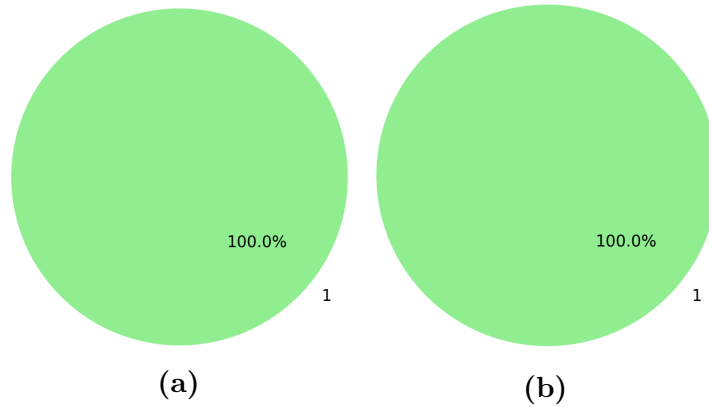


Figure 6.20: People found at the end of all runs. Classes correspond to the number of people detected. (a) Scenario DANGER; (b) Scenario DISCOVER.



Figure 6.21: Actions required to detect people found vs. total number of actions. The red dotted line is placed in correspondence with the mean of the actions required to detect people found, while the purple one is placed to represent the mean of the total number of actions executed. (a) Scenario DANGER; (b) Scenario DISCOVER.

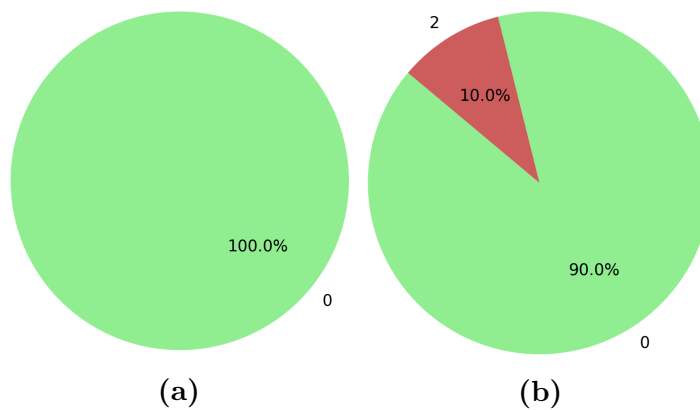


Figure 6.22: Minimum number of actions required to find the people found for all runs. Classes correspond to the number of people detected. (a) Scenario DANGER; (b) Scenario DISCOVER.

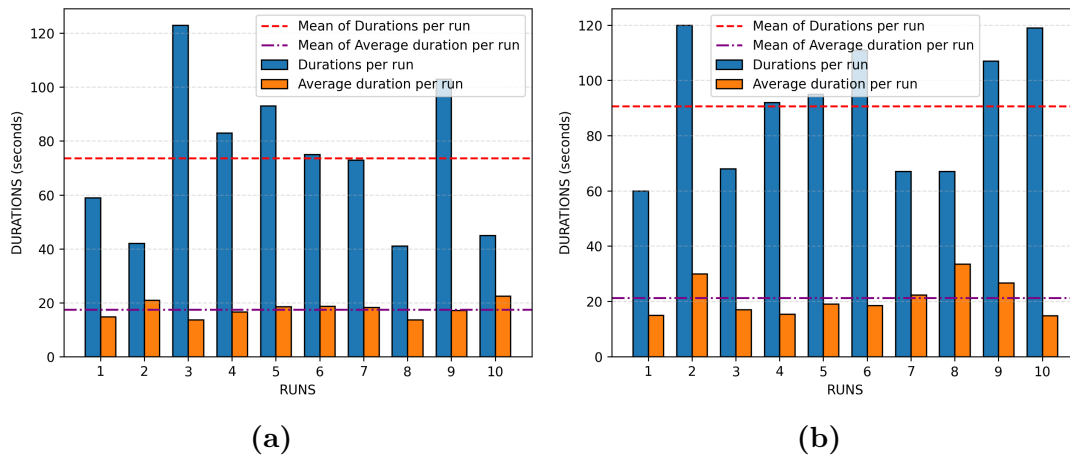


Figure 6.23: Durations per run vs. average duration per run. The red dotted line is placed in correspondence with the mean of the durations per run, while the purple one is placed to represent the mean of the average duration per run. (a) Scenario DANGER; (b) Scenario DISCOVER.

Figure 6.24 and Figure 6.25 are strictly related because they refer to the same test run in a given scenario. The first figure shows the evolution of people detection confidence over time and robot-people distance: black and empty circles have to become black and filled (since UNDEFINED points have to become LIKELY points) until some stars appear if the robot gets closer to people, otherwise they have to vanish. Moreover, the detection in the DANGER scenario have higher confidence, even though they are few since there is only one person that can be found. However, in the other scenario, the increase in confidence is very visible (e.g.: the levels of confidence between the cyan star and the red star in Figure 6.24(b)). This is related to the discovery behavior of the robot, which is displayed also in Figure 6.25. This figure shows the evolution of the path of the robot in blue, where the direction is indicated by the black arrow. People are still represented by stars. The motion of the robot is quite similar in both figures, but on the right one, the robot stops between the obstacle and the person, while on the left figure, it stops next to the person, where the small desk is less visible. These different behaviors can be motivated by explaining that in the DANGER scenario the robot focuses mainly on people, while on the other one it has to achieve poses that could let it detect something more. This is another test that provides proof of the correctness of our hypothesis about the change in robot behavior according to the scenario.

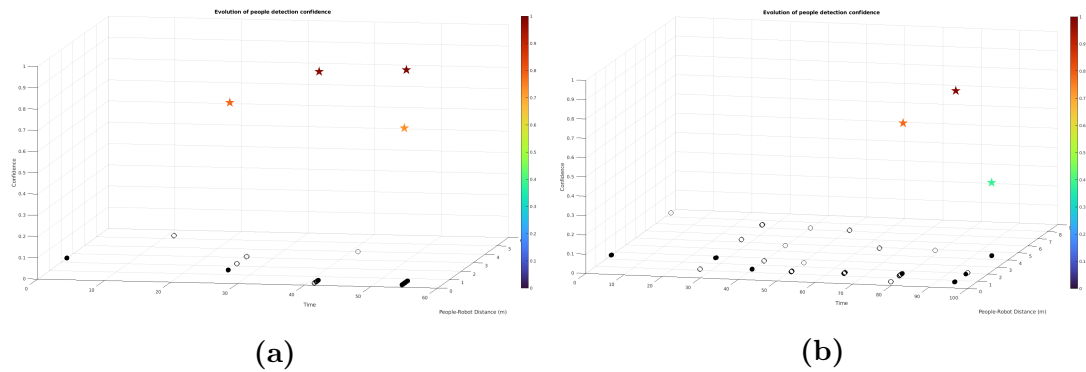


Figure 6.24: Test: occlusion to target. Comparison of the evolution of people detection confidence over time and robot-people distance. Black and empty circles are the points labeled as UNDEFINED. Black and filled circles are the LIKELY points. Stars represent people detected at a given time and robot-people distance. The color scale on the right represents the confidence level associated with a person detection. (a) Scenario DANGER; (b) Scenario DISCOVER.

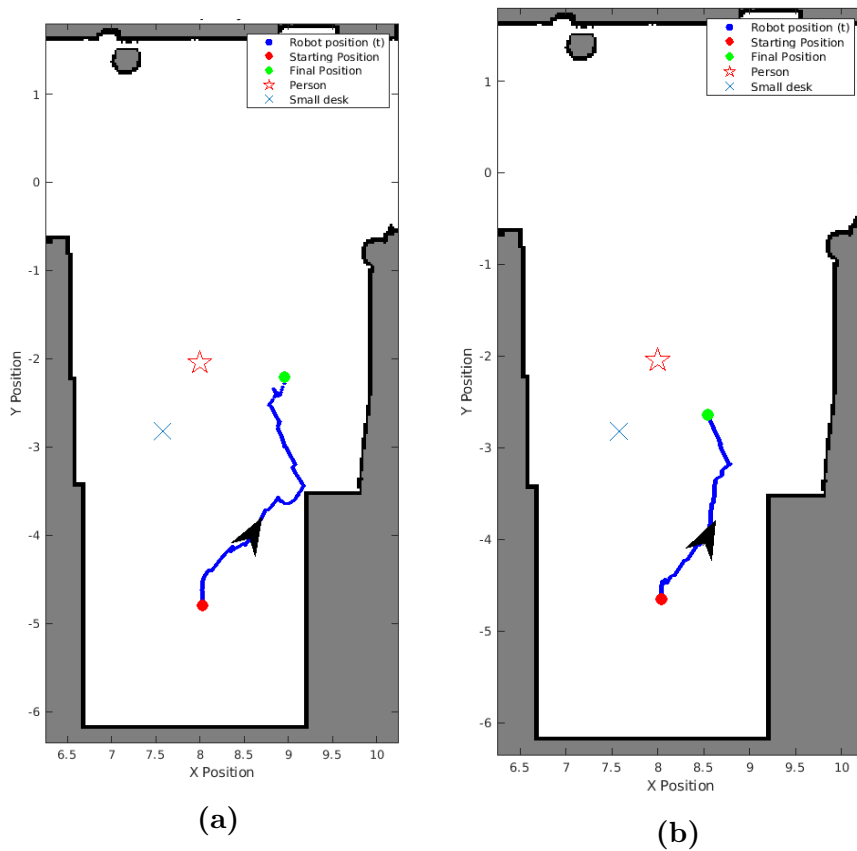


Figure 6.25: Test: occlusion to target. A blue trajectory is the path of the robot during a test run. The black arrows indicate the direction of motion. The red and green points are the starting and the final positions. Stars represent people. (a) Scenario DANGER; (b) Scenario DISCOVER.

6.4.5 Results on occlusion to object

The results in the following figures are very similar, it does not matter the scenario. This test has been performed to check the behavior of the robot in a context without people, which is the opposite of a crowded environment. No people are detected, hence no false positives were found (Figures 6.26, Figure 6.27). Also, the mean durations of actions are similar (Figure 6.23).

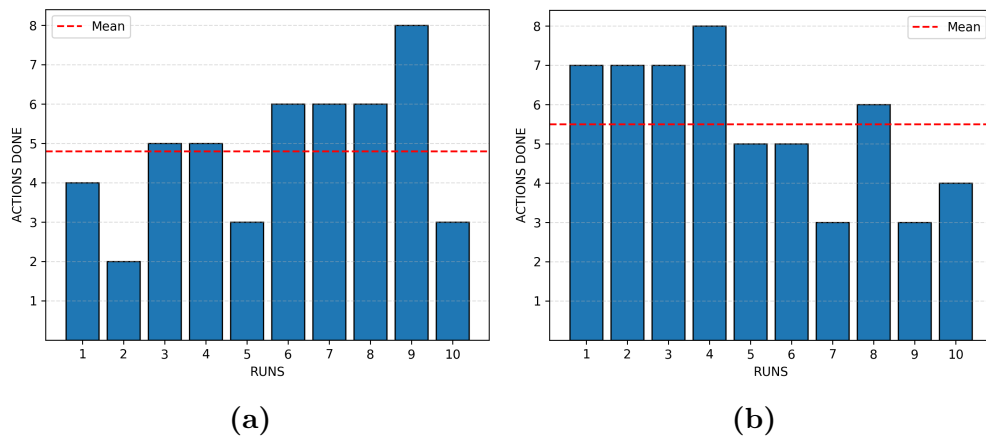


Figure 6.26: Comparison of people found in the ten runs. The red dotted line is placed in correspondence with the mean of the people found. in this test there are no people. (a) Scenario DANGER; (b) Scenario DISCOVER.

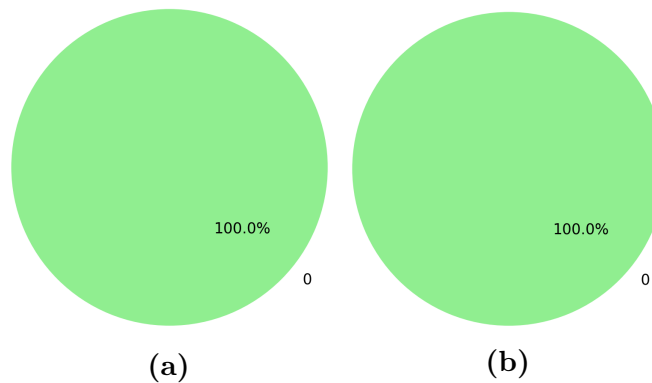


Figure 6.27: People found at the end of all runs. Classes correspond to the number of people detected. (a) Scenario DANGER; (b) Scenario DISCOVER.

Since in this test there are no people involved, the resulting plot about confidence evolution is not presented because it is not enough significant. However, Figure 6.29 shows the path of the robot in blue, where the direction is indicated by the black arrow. The small desk and the pole are represented by the cross and the circle. In this test, we tried to confuse the robot about the pole as one of two possible human legs. In Figure 6.29(a) the robot detects no people, hence its actions are very short, while in Figure (b) the robot is working in the DISCOVER

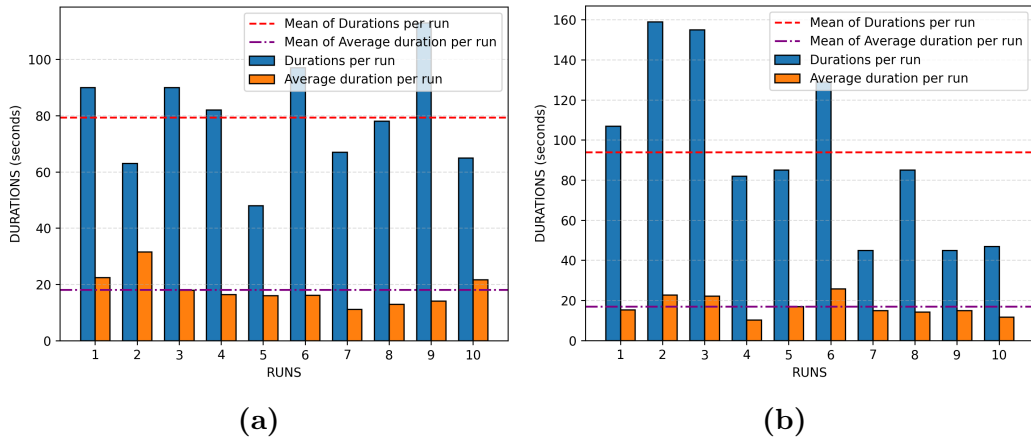


Figure 6.28: Durations per run vs. average duration per run. The red dotted line is placed in correspondence with the mean of the durations per run, while the purple one is placed to represent the mean of the average duration per run. (a) Scenario DANGER; (b) Scenario DISCOVER.

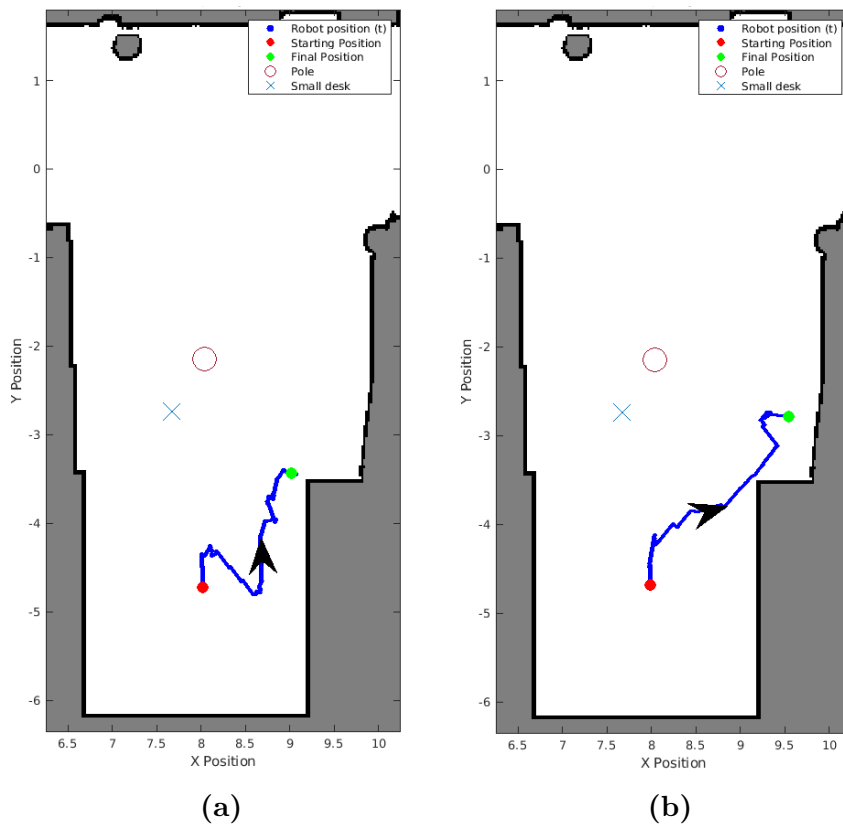


Figure 6.29: Test: occlusion to object. A blue trajectory is the path of the robot during a test run. The black arrows indicate the direction of motion. The red and green points are the starting and the final positions. Stars represent people. (a) Scenario DANGER; (b) Scenario DISCOVER.

scenario, so it moves around more than before. This test provides proof again of the correctness of our hypothesis about the change in robot behavior according to the scenario.

6.4.6 Results on double target occlusion with objects

The results in the following figures are good for both scenarios. The occluded person is always detected very soon (Figures 6.30, Figure 6.31). In the DISCOVER scenario, fewer actions are performed, but the first detection come a bit later than in the DANGER scenario. This is allowed because of the working principle (Figure 6.32, Figure 6.33). Mean durations of actions are more or less the same, but they seem to be more regular in the DANGER case (Figure 6.34). Hence, since in this test the only person is occluded by two objects (the small desk and the pole), the robot could be stressed but it behaves correctly.

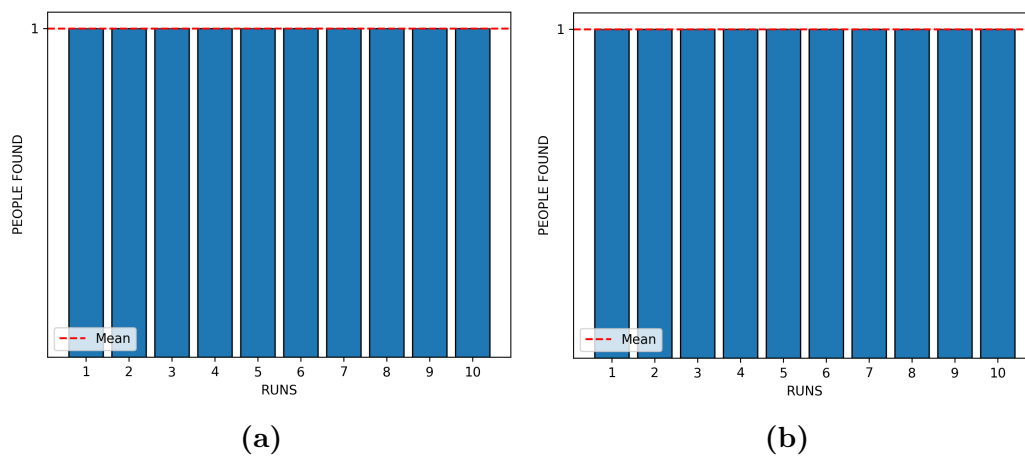


Figure 6.30: Comparison of people found in the ten runs. The red dotted line is placed in correspondence with the mean of the people found. Real number of people in the test is one. (a) Scenario DANGER; (b) Scenario DISCOVER.

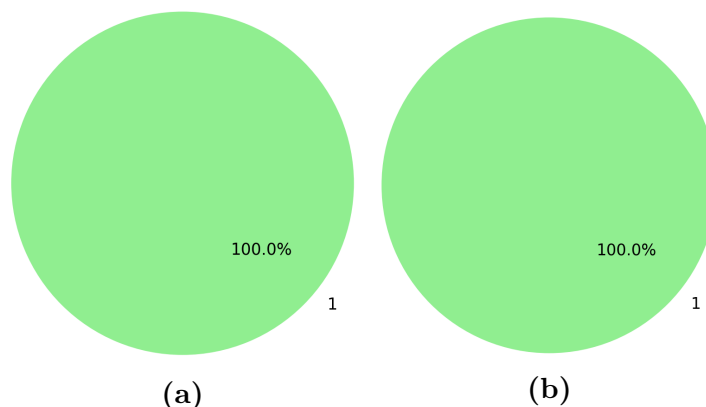


Figure 6.31: People found at the end of all runs. Classes correspond to the number of people detected. (a) Scenario DANGER; (b) Scenario DISCOVER.

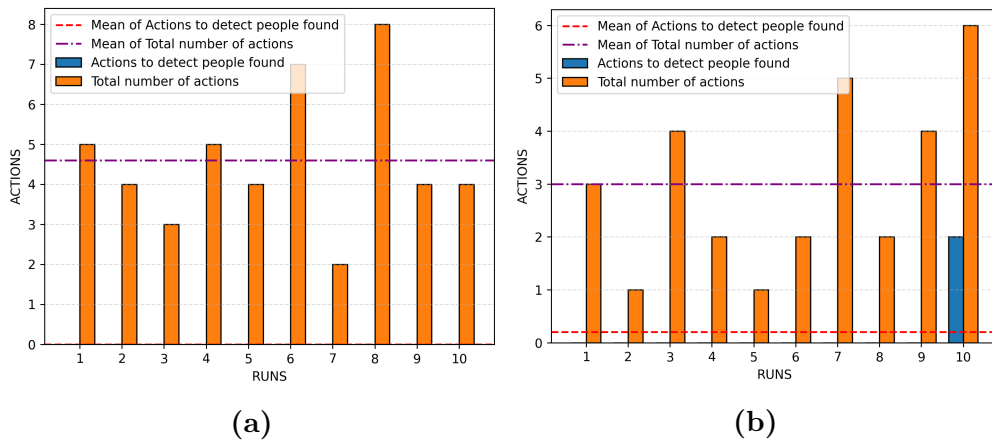


Figure 6.32: Actions required to detect people found vs. total number of actions. The red dotted line is placed in correspondence with the mean of the actions required to detect people found, while the purple one is placed to represent the mean of the total number of actions executed. (a) Scenario DANGER; (b) Scenario DISCOVER.

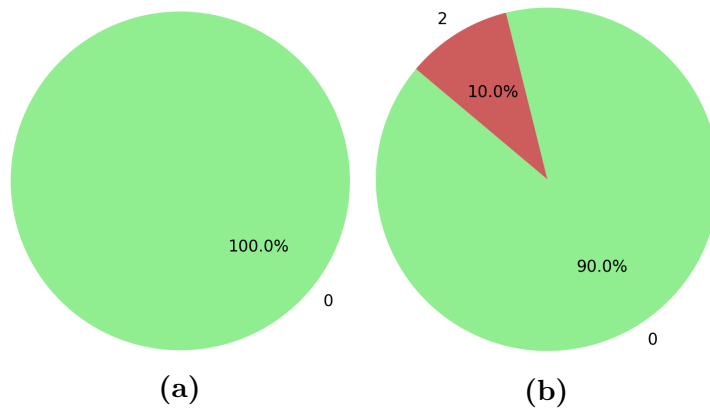


Figure 6.33: Minimum number of actions required to find the people found for all runs. Classes correspond to the number of people detected. (a) Scenario DANGER; (b) Scenario DISCOVER.

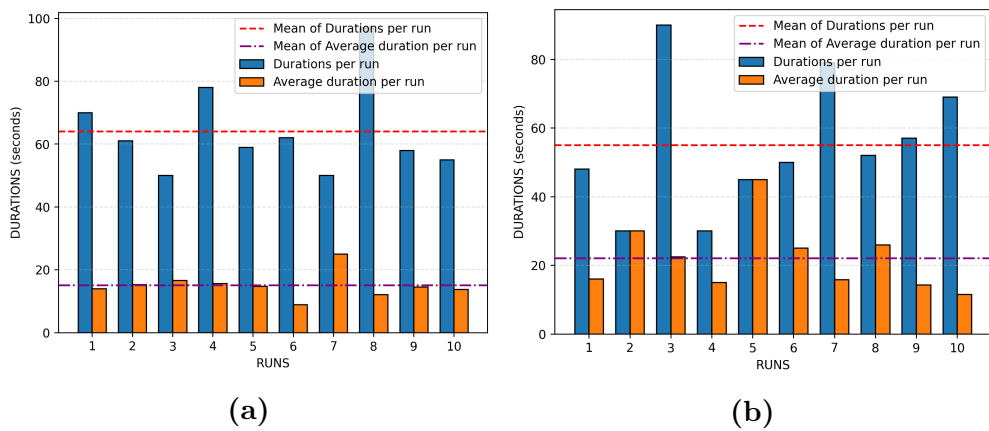


Figure 6.34: Durations per run vs. average duration per run. The red dotted line is placed in correspondence with the mean of the durations per run, while the purple one is placed to represent the mean of the average duration per run. (a) Scenario DANGER; (b) Scenario DISCOVER.

Figure 6.35 and Figure 6.36 are strictly related because they refer to the same test run in a given scenario. The first figure shows the evolution of people detection confidence over time and robot-people distance: black and empty circles have to become black and filled (since UNDEFINED points have to become LIKELY points) until some stars appear if the robot gets closer to people, otherwise they have to vanish. Moreover, the detection in the DANGER scenario have higher confidence that tends to increase on average, even though they are few since there is only one person that can be found. However, in the other scenario, the increase in confidence is bigger. This is related to the discovery behavior of the robot, which is displayed also in Figure 6.36. This figure shows the evolution of the path of the robot in blue, where the direction is indicated by the black arrow. People are still represented by stars. In this test, there is the small desk that occludes one leg of the human, while the pole adds noise to a possible detection of the other leg. The motion of the robot is very different in the two cases: on the left, it moves to increase visibility on the human, but on the right figure, the robot stops next to the desk. This is due to a starting uncertain detection in the DANGER scenario, indeed the path is not smoothed since the robot always detects only that person. In the other scenario, the starting detection of the human has middle-high confidence, which is enough for the robot to be sure of the human, hence it moves with a discovery behavior. The final position lets the robot check better the perimeter of the desk so that no new people are found. This stress test provides notable information about the correct execution of our work.

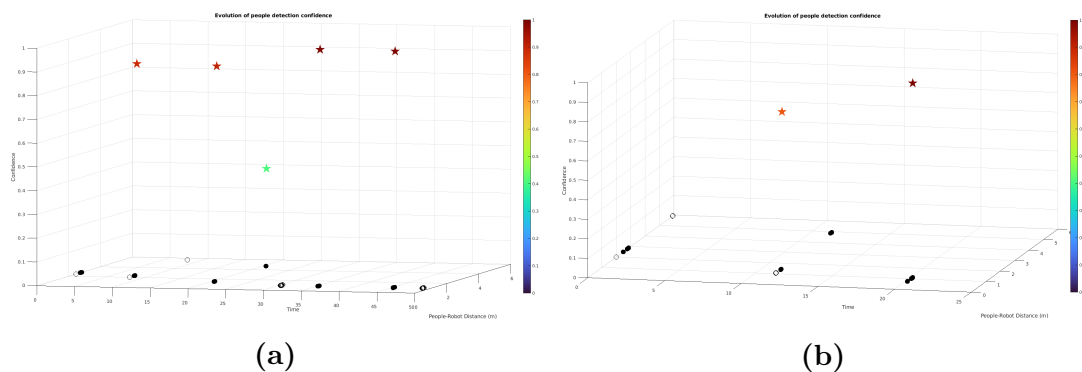


Figure 6.35: Test: double occlusion to target. Comparison of the evolution of people detection confidence over time and robot-people distance. Black and empty circles are the points labeled as UNDEFINED. Black and filled circles are the LIKELY points. Stars represent people detected at a given time and robot-people distance. The color scale on the right represents the confidence level associated with a person detection. (a) Scenario DANGER; (b) Scenario DISCOVER.

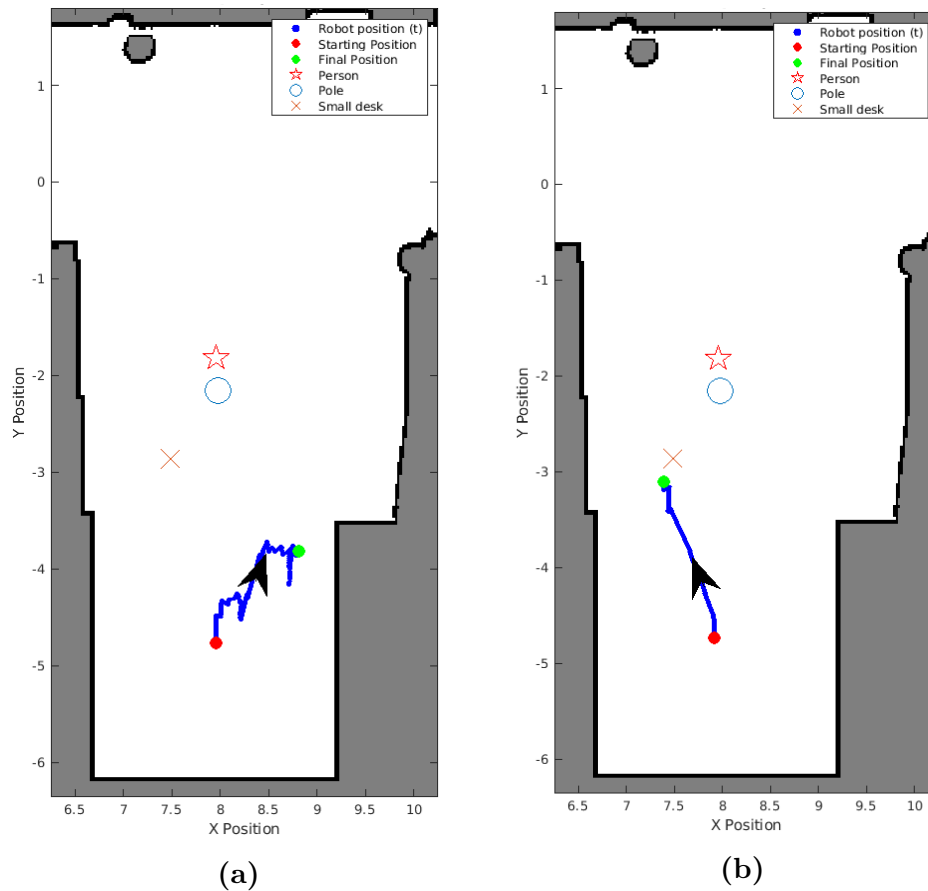


Figure 6.36: Test: double occlusion to target. A blue trajectory is the path of the robot during a test run. The black arrows indicate the direction of motion. The red and green points are the starting and the final positions. Stars represent people. (a) Scenario DANGER; (b) Scenario DISCOVER.

6.4.7 Accuracy comparison

In Figure 6.37 the comparison between the accuracies achieved in each test is shown, according to the scenario. The mean accuracy for the DANGER scenario is higher than the one for the DISCOVER. This verifies the hypothesis about the use of active perception to behave according to the scenario, to improve the robot's confidence about the environment and general detection, as people detection can be in particular.

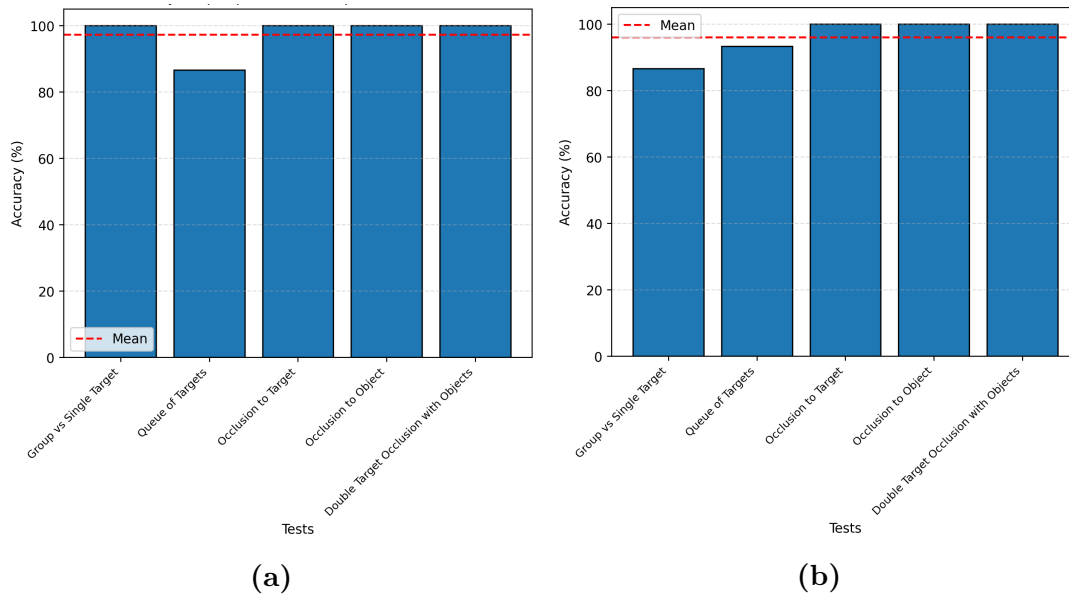


Figure 6.37: Comparison of the accuracies achieved in all the tests performed. The red dotted line is placed in correspondence with the mean accuracy. (a) Scenario DANGER; (b) Scenario DISCOVER.

6.4.8 Correlation over data

As anticipated in Section 6.3, the final analysis of test data collected regards a possible correlation over the variables people-robot distances and people detection confidences.

The data collected in Table 6.1 and the two figures below have been computed with MATLAB. Figure 6.38 shows the correlation over the variables in the DANGER scenario, while Figure 6.39 shows the one for DISCOVER.

Data	Scenario <i>DANGER</i>	Scenario <i>DISCOVER</i>
N.° Detection	165	160
N.° Distances	165	160
N.° Confidences	165	160
Linear regression coefficients	(-0.039811, 0.97003)	(-0.045956, 0.96382)
Correlation	-0.34129	-0.35014

Table 6.1: Summary table with analysis on correlation over people-robot distances and people detection confidences. The correlation indices are in bold. The linear regression coefficients are the angular coefficient m_{lr} and the intercept q_{lr} of the linear regression line.

In the table above, the numbers of detection, distances, and confidences are equal since they are the components of a robot detection done in a precise time and at a certain distance from the people with a level of confidence. Moreover, this comparison between scenarios can be performed since the numbers of samples are very similar.

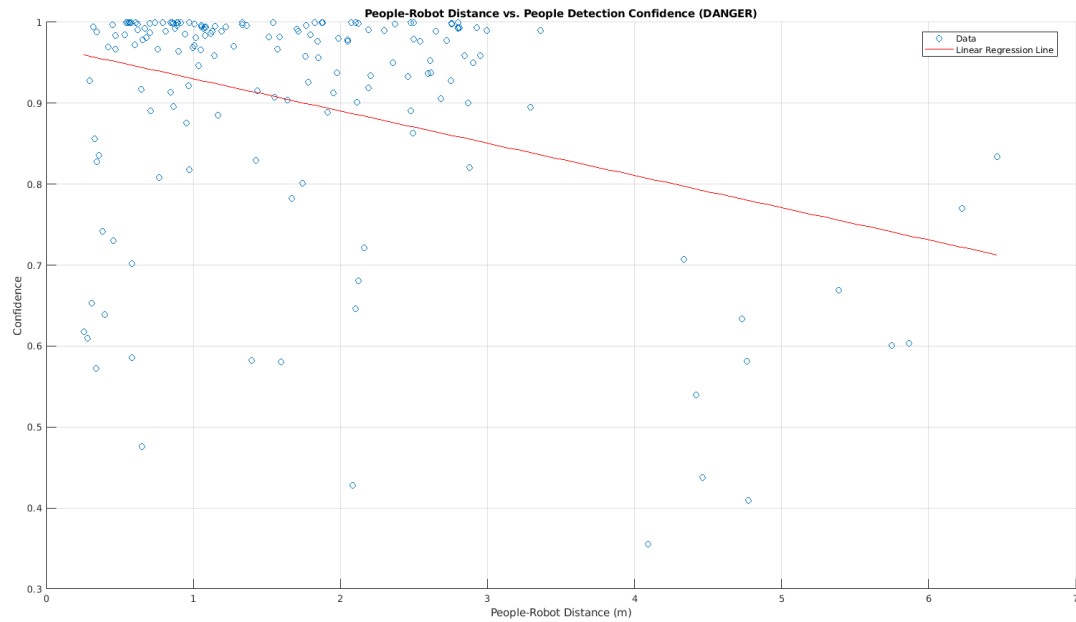


Figure 6.38: Plot of the data collected in the DANGER scenario. Little blue circles are the people detection done at a certain distance. The linear regression correlation line is the one in red.

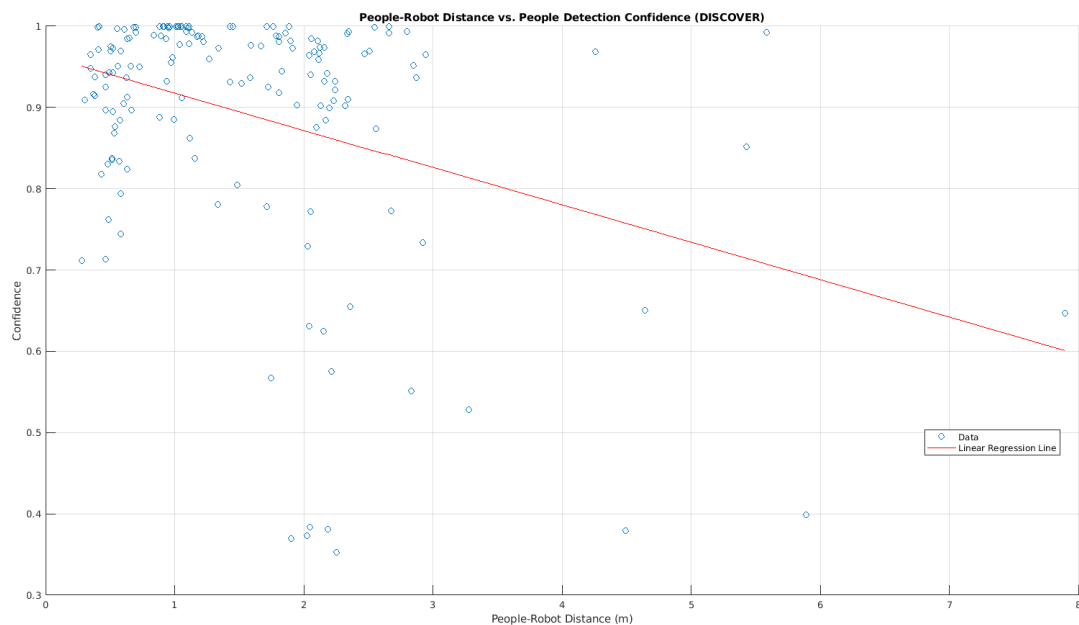


Figure 6.39: Plot of the data collected in the DISCOVER scenario. Little blue circles are the people detection done at a certain distance. The linear regression correlation line is the one in red.

The final regressions confirm the correlation over the two variables. In both cases, there is a negative correlation, which means the increase of one variable is followed by the decrease of the other one, coherently with our hypothesis. For sure the acquisition of more data can help reveal more about this relationship.

6.5 Assessment Questionnaire

We administrated a questionnaire to all the ten people (seven boys and three girls) who took part in the test at the Autonomous Robotics Laboratory of the University of Padova. Only three of them had previous experience with real robots.

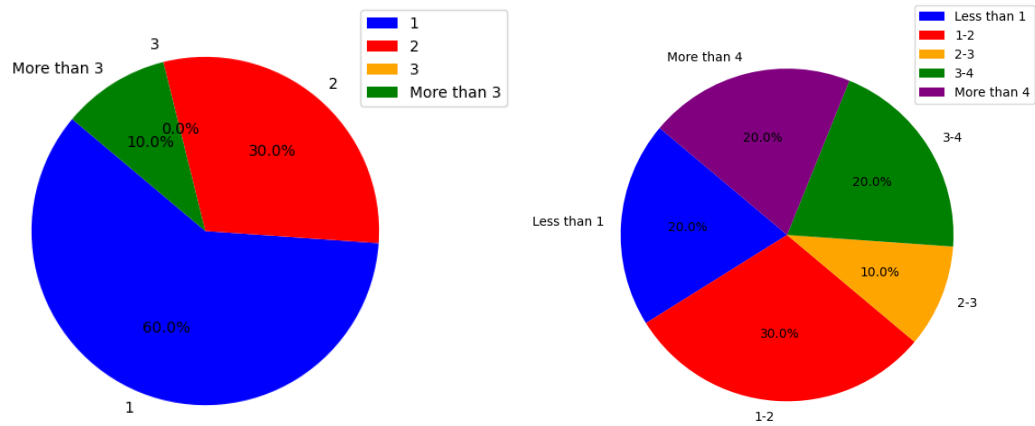
The questionnaire is composed of eight simple questions to collect feedback both using numerical indices and short open questions. The information about the eight questions have been resumed in Table 6.2, while the exact questions are listed as follows:

1. *Q1*: How many times have you participated to this activity?
2. *Q2*: How many total hours were you involved in the experience?
3. *Q3*: With respect to the definitions in the image above (Figure 6.42), evaluate the degree of compliance with the most common sociability indices about the robot's behavior during the tests
4. *Q4*: How do you evaluate the robot, the tools, and the equipment available in terms of work functionality? Please refer to possible limitations about the application of these tools in real world for this research topic
5. *Q5*: Do you consider the robot has been able to detect you?
6. *Q6*: What would you like the robot to do as soon as it detects you?
7. *Q7*: Do you have fear about the robot's behavior?
8. *Q8*: Do you have any other general comment on the experience?

Number	Type	Values	References
Q1	Closed	[0, 1, 2, More than 3]	-
Q2	Closed	[Less than 1, 1-2, 2-3, 3-4, More than 4]	-
Q3	Closed	[1,2, 3, 4, 5, 6, 7, 8]	-
Q4	Closed	[1, 2, 3, 4, 5]	[30]
Q5	Open	-	-
Q6	Open	-	-
Q7	Open	-	-
Q8	Open	-	-

Table 6.2: Table of metrics displayed in the histograms in Figure 5.12 and in Figure 5.13.

The distributions of the answers to questions Q1, Q2, and Q4 are reported in Figure 6.40 and in Figure 6.41.



(a) Distribution of the answers for Q1. (b) Distribution of the answers for Q2.

Figure 6.40: Distribution of the answers for the first two questions Q1 and Q2. The legends collect the possible answers to each question.

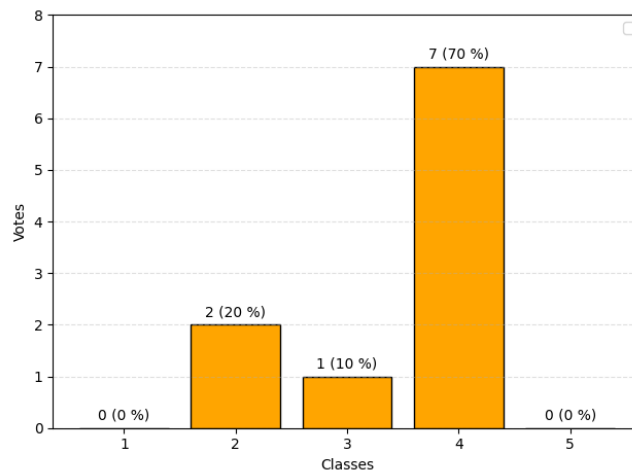


Figure 6.41: Distribution of the answers for Q4.

The answers to Q5 are:

- Yes;
- Sometimes;
- Most of the times, yes;
- Sometimes;
- Yes I do;

- Yes, it is;
- Most of the times.

The answers to Q6 are:

- Play a musical note or shake its arm;
- Say hello to me in some way or make a noise if possible;
- Make it clear that it sees me, for example via an audio cue;
- For instance greet me;
- acknowledge me in some way and keep on with its task;
- Raise his arms and say Hello;
- The robot should do a sound cue.

The answers to Q7 are:

- No;
- Not at all;
- No, the robot was moving very slowly and it did not cause fear;
- Nothing;
- Most of the times no;
- No I do not;
- Absolutely no;
- The robot does not cause fear.

The answers to Q8 are:

- A very interesting experiment, I would have liked to see it in a larger space;
- I think the robot is a bit too large and it would need more space to move better;
- The robot should take much space among the person;

- You should perform tests in a larger space with more people.

The questions above refer to practical information, while question Q3 asks the evaluation of some of the most common sociability indices. Their graphical representation, which we also included in the questionnaire, can be found in Figure 6.42 proposed in [30]. Instead, the distribution of the answers for Q3 is in Figure 6.43.

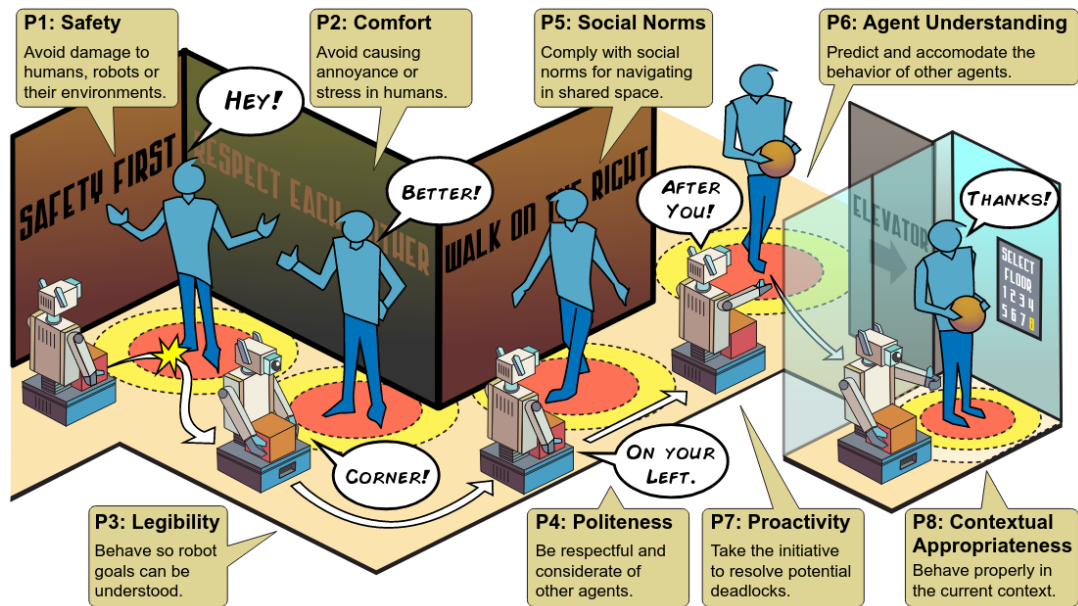


Figure 6.42: Most popular social navigation indices that can be computed to evaluate the proper work. They have been proposed in [30].

Given the small sample size of the answers, no statistical tests were performed.

The people in the experiments have been able to understand when the robot detected them since it moved properly. They had no fear about the robot's behavior, rather they would like to receive feedback from the robot itself, like a sound cue. Finally, the most shared tip is to test our approach in a big environment, which is the final use case of the work for this thesis.

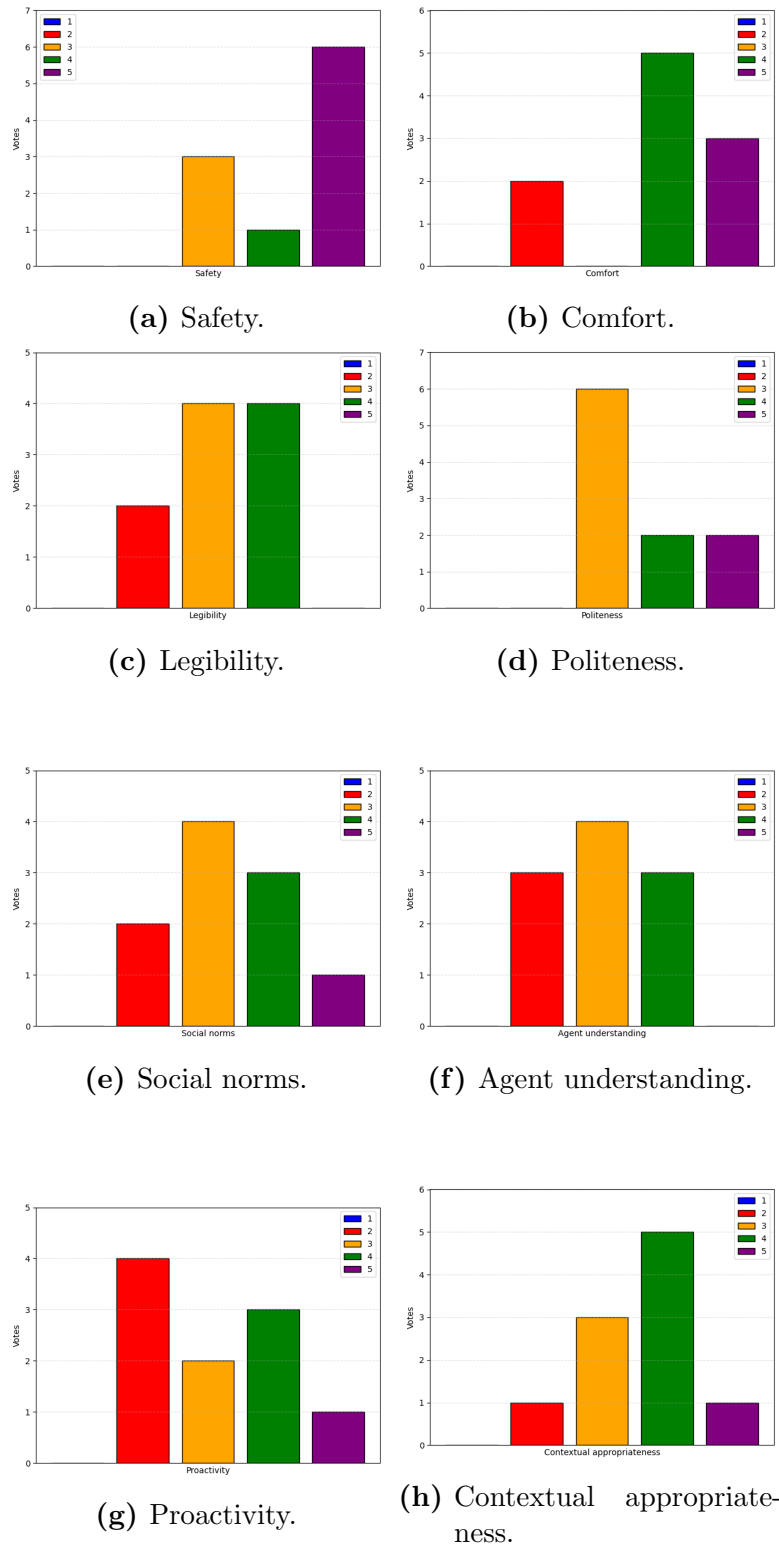


Figure 6.43: Distributions of the answers for the social indices listed in the questionnaire and proposed in [30]. Possible values for the indices go from 1 to 5. The colors associated to them are blue, red, orange, green, and purple.

Chapter 7

Conclusions

In this final chapter, the conclusion about our work and the future works we propose are described.

7.1 Discussions

The goal of this thesis was to develop at least one possible solution to the active perception task, to perform in crowded environments, to solve the problem of people detection. In this thesis, the focus was not on improving the detection module, but on making the robot autonomous in moving around the environment to achieve this goal, exploiting only data incoming from laser sensor data.

We worked mainly on a policy behavior to guide the robot in its perception task, in an active way, differentiating two types of scenarios: “DANGER” and “DISCOVER”. In the former, the robot must pay more attention to perceiving and act to find more people as possible and in less time it can; while in the latter, the robot is allowed to move around the environment with the same final goal (e.g.: to detect more people it can), but without imposing too many constraints on time and trajectory.

Simulations of this approach ran in dynamic environments, while we carried on the tests in a static environment to check better how the robot moved, according to what it was able to sense.

The deployment on the TIAGo++ robot was done at the Autonomous Robotics Laboratory of the University of Padova: the robot started from a position and then it was free to navigate in the test environment to increase confidence about people, hence true positive, and to decrease confidence about false positive objects and obstacles. These initial real-world tests proved this approach to work

well, both in a qualitative and quantitative evaluation. Indeed, the robot could detect people in the scene and behave according to the particular scenario in which it has to work.

People involved in the experiments suggested proving this approach in a larger environment so that problems of space can be solved, and the robot can approach people over their intimate space. We expected to get this feedback since, because of restricted space, we used a small space to get a more crowded context as possible.

Active perception is a difficult task. Human behaviors are different among people because they are influenced by the context in which the human is living, and on the current mood, emotions, and physical state. For the robot, instead, both simulated and real-world navigation are one of the weaknesses. In some runs of the tests, the robot does not follow a precise path and recomputes its final goal because of delocalization, mainly in real-world use, which is of fundamental importance to match the robot's position in the environment, objects, people, and the environment itself. However, for the aim of this thesis, the navigation module considered is the one provided by ROS and the developers of TIAGo++ robot. Finally, the robot has limited memory, computational resources, and battery, hence developing methods that must not be much complex and time-resources consuming, increases these difficulties.

7.2 Future works

Future works will be focused on improving the performances by integrating a module for the check of the goal pose that the robot can choose among a set of proposed goal poses, considering also their feasibility in the environment and the context.

The next steps could be a deeper analysis and validation of the other two methods we proposed for learning the policy using DNN and RL. The former method should be able to decrease the computation done by the policy method, while the latter should provide an approach feasible for real-time action execution, to let the robot work in a highly dynamic world by adapting in real-time its actions.

The next tests should consider larger environments, both static and dynamic, as gathered from the questionnaire answers. Moreover, when the robot detects and approaches people next to its final goal pose, it should send some cues (e.g.:

a short sound cue or a little motion of his arm).

In conclusion, the future of active perception for the detection of people or for particular object recognition can supply another working direction for the improvement of current state-of-the-art methods.

References

- [1] Ladji Adiaviakoye, Plainchault Patrick, Bolircene Marc, and Jean-Michel Auberlet. Tracking of multiple people in crowds using laser range scanners. In *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6, 2014.
- [2] Eugenio Aguirre and Miguel Garcia-Silvente. Detecting and tracking using 2d laser range finders and deep learning. *Neural Computing and Applications*, 35, 09 2022.
- [3] Emmanuel Alao and Philippe Martinet. Uncertainty-aware navigation in crowded environment. In *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 293–298, 2022.
- [4] Silas F. R. Alves, Alvaro Uribe-Quevedo, Delun Chen, and Jon Morris. Developing a vr socially assistive robot simulator employing game development tools. In *2022 IEEE Games, Entertainment, Media Conference (GEM)*, pages 1–4, 2022.
- [5] Javad Amirian, Bingqing Zhang, Francisco Valente Castro, Juan Jose Balde-lomar, Jean Bernard Hayet, and Julien Pettre. Opentraj: Assessing prediction complexity in human trajectories datasets. *CoRR*, abs/2010.00890, 2020.
- [6] Fernando Amodeo, Noé Pérez-Higueras, Luis Merino, and Fernando Ca-ballero. Frog: A new people detection dataset for knee-high 2d range finders, 2023.
- [7] Akash Arora, P. Michael Furlong, Robert Fitch, Salah Sukkariéh, and Terrence Fong. Multi-modal active perception for information gathering in sci-ence missions, 2017.

-
- [8] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.
- [9] Alberto Bacchin, Gloria Beraldo, and Emanuele Menegatti. Learning to plan people-aware trajectories for robot navigation: A genetic algorithm *. pages 1–6, 08 2021.
- [10] Ruzena Bajcsy, Yiannis Aloimonos, and John K. Tsotsos. Revisiting active perception, 2016.
- [11] Luca Bartolomei, Lucas Teixeira, and Margarita Chli. Semantic-aware active perception for uavs using deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3101–3108, 2021.
- [12] Nicola Bellotto and Huosheng Hu. A bank of unscented kalman filters for multimodal human perception with mobile service robots. *I. J. Social Robotics*, 2:121–136, 06 2010.
- [13] Graeme Best, Jan Faigl, and Robert Fitch. Online planning for multi-robot active perception with self-organising maps. *Autonomous Robots*, 42, 04 2018.
- [14] Lucas Beyer, Alexander Hermans, Timm Linder, Kai O. Arras, and Bastian Leibe. Deep person detection in 2d range data, 2018.
- [15] Rashmi Bhaskara, Maurice Chiu, and Aniket Bera. Sg-lstm: Social group lstm for robot navigation through dense crowds, 2023.
- [16] Manuel Boldrer, Alessandro Antonucci, Paolo Bevilacqua, Luigi Palopoli, and Daniele Fontanelli. Multi-agent navigation in human-shared environments: A safe and socially-aware approach. *Robotics and Autonomous Systems*, 149:103979, 12 2021.
- [17] Kuanqi Cai, Weinan Chen, Chaoqun Wang, Shuang Song, and Max Meng. Human-aware path planning with improved virtual doppler method in highly dynamic environments. *IEEE Transactions on Automation Science and Engineering*, PP:1–18, 01 2022.

- [18] Jiyu Cheng, Hu Cheng, Max Meng, and Hong Zhang. Autonomous navigation by mobile robots in human environments: A survey. pages 1981–1986, 12 2018.
- [19] Lindsey Coffee-Johnson and Debbie Perouli. Detecting anomalous behavior of socially assistive robots in geriatric care facilities. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 582–583, 2019.
- [20] Sara Cooper and Séverin Lemaignan. Towards using behaviour trees for long-term social robot behaviour. In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 737–741, 2022.
- [21] Javier Correa and Álvaro Soto. Active visual perception for mobile robot localization. *Journal of Intelligent and Robotic Systems*, 58:339–354, 2010.
- [22] Serhan Coşar and Nicola Bellotto. Human re-identification with a robot thermal camera using entropy-based sampling. *Journal of Intelligent & Robotic Systems*, 98, 04 2020.
- [23] Yuxiang Cui, Xiaolong Huang, Yue Wang, and Rong Xiong. Socially-aware multi-agent following with 2d laser scans via deep reinforcement learning and potential field. 09 2021.
- [24] Yuxiang Cui, Haodong Zhang, Yue Wang, and Rong Xiong. Learning world transition model for socially aware robot navigation. pages 9262–9268, 05 2021.
- [25] Daniel Dugas, Kuanqi Cai, Olov Andersson, Nicholas Lawrance, Roland Siegwart, and Jen Jen Chung. Flowbot: Flow-based modeling for robot navigation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8799–8805, 2022.
- [26] Juan Fasola and Maja J Mataric. Using socially assistive human–robot interaction to motivate physical exercise for older adults. *Proceedings of the IEEE*, 100(8):2512–2526, 2012.
- [27] Gonzalo Ferrer, Anais Garrell, and A. Sanfeliu. Robot companion: A social-force based approach with human awareness-navigation in crowded environments. pages 1688–1694, 11 2013.

- [28] Gonzalo Ferrer, Anaís Garrell, and Alberto Sanfeliu. Robot companion: A social-force based approach with human awareness-navigation in crowded environments. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1688–1694, 2013.
- [29] Amalia Foka and Panos Trahanias. Probabilistic autonomous robot navigation in dynamic environments with human motion prediction. *I. J. Social Robotics*, 2:79–94, 03 2010.
- [30] Anthony Francis, Claudia Pérez-D’Arpino, Chengshu Li, Fei Xia, Alexandre Alahi, and Rachid Alami et al. Principles and guidelines for evaluating social robot navigation algorithms. 2023.
- [31] T. Germa, F. Lerasle, Nouredine Ouadah, and Viviane Cadenat. Vision and rfid data fusion for tracking people in crowds by a mobile robot. *Computer Vision and Image Understanding*, 114:641–651, 06 2010.
- [32] Óscar Gil, Anaís Garrell, and Alberto Sanfeliu. Social robot navigation tasks: Combining machine learning techniques and social force model. *Sensors*, 21(21), 2021.
- [33] Dylan F. Glas, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. Simultaneous people tracking and localization for social robots using external laser range finders. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 846–853, 2009.
- [34] Ángel Guerrero-Higueras, Claudia Álvarez Aparicio, Maria Carmen Olivera, Francisco Rodríguez Lera, Camino Fernández, Francisco Martín, and Vicente Matellán. Tracking people in a mobile robot from 2d lidar scans using full convolutional neural networks for security in cluttered environments. *Frontiers in Neurorobotics*, 12:85, 01 2019.
- [35] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: socially acceptable trajectories with generative adversarial networks. *CoRR*, abs/1803.10892, 2018.
- [36] Marc Hanheide, Annika Peters, and Nicola Bellotto. Analysis of human-robot spatial behaviour applying a qualitative trajectory calculus. pages 689–694, 09 2012.

- [37] Kensuke Harada, Tokuo Tsuji, Soichiro Uto, Natsuki Yamanobe, Kazuyuki Nagata, and Kosei Kitagaki. Stability of soft-finger grasp under gravity. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 883–888, 2014.
- [38] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical Review E*, 51, 05 1998.
- [39] Jarrett Holtz and Joydeep Biswas. Socialgym: A framework for benchmarking social robot navigation, 2022.
- [40] Jongmin Jeong, Tae Yoon, and Jin Park. Towards a meaningful 3d map using a 3d lidar and a camera. *Sensors*, 18:2571, 08 2018.
- [41] Dan Jia, Alexander Hermans, and Bastian Leibe. DR-SPAAM: A Spatial-Attention and Auto-regressive Model for Person Detection in 2D Range Data. In *International Conference on Intelligent Robots and Systems (IROS)*, 2020.
- [42] Dan Jia, Mats Steinweg, Alexander Hermans, and Bastian Leibe. Self-supervised person detection in 2d range data using a calibrated camera, 2021.
- [43] Dan Jia, Mats Steinweg, Alexander Hermans, and Bastian Leibe. Self-Supervised Person Detection in 2D Range Data using a Calibrated Camera. 2021.
- [44] Vishnu K. Narayanan, Anne Spalanzani, Francois Pasteau, and Marie Babel. On equitably approaching and joining a group of interacting humans. pages 4071–4077, 09 2015.
- [45] Burak Kaleci, Kaya Turgut, and Helin Dutağacı. 2dlasernet: A deep learning architecture on 2d laser scans for semantic classification of mobile robot locations. *Engineering Science and Technology, an International Journal*, 28, 06 2021.
- [46] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.
- [47] Mincheul Kim, Youngsun Kwon, and Sung-Eui Yoon. Group estimation for social robot navigation in crowded environments. In *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*, pages 1421–1425, 2022.

- [48] Hasan Kivrak, Furkan Çakmak, Hatice Kose, and Sirma Yavuz. Waypoint based path planner for socially aware robot navigation. *Cluster Computing*, 25, 06 2022.
- [49] Hasan Kivrak, Furkan Çakmak, Hatice Kose, and Sirma Yavuz. Waypoint based path planner for socially aware robot navigation. *Cluster Computing*, 25, 06 2022.
- [50] Pavlina Konstantinova, Alexander Udvardy, and Tzvetan Semerdjiev. A study of a target tracking algorithm using global nearest neighbor approach. In *Proceedings of the 4th International Conference Conference on Computer Systems and Technologies: E-Learning, CompSysTech '03*, page 290–295, New York, NY, USA, 2003. Association for Computing Machinery.
- [51] Parth Kothari, Sven Kreiss, and Alexandre Alahi. Human trajectory forecasting in crowds: A deep learning perspective. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):7386–7400, 2022.
- [52] Steven LaValle and James Kuffner. Randomized kinodynamic planning. *I. J. Robotic Res.*, 20:378–400, 01 2001.
- [53] Hong Thai Le, Duy Thao Nguyen, and Xuan Tung Truong. Socially aware robot navigation framework in crowded and dynamic environments: A comparison of motion planning techniques. In *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 95–101, 2021.
- [54] Min-Fan Ricky Lee and Sharfiden Hassen Yusuf. Mobile robot navigation using deep reinforcement learning. *Processes*, 10(12), 2022.
- [55] Angus Leigh, Joelle Pineau, Nicolas Olmedo, and Hong Zhang. Person tracking and following with 2d laser scanners. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 726–733, 2015.
- [56] Shengming Li, Lin Feng, Yunfei Ge, Li Zhu, and Liang Zhao. An ensemble learning method for robot electronic nose with active perception. *Sensors*, 21(11), 2021.
- [57] Ela Liberman-Pincu and Tal Oron-Gilad. Exploring the effect of mass customization on user acceptance of socially assistive robots (sars). In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 880–884, 2022.

-
- [58] Dario Mammolo. Active slam in crowded environments. Master thesis, ETH Zurich, Zurich, 2019.
- [59] Christoforos Mavrogiannis, Francesca Baldini, Allan Wang, Dapeng Zhao, Pete Trautman, Aaron Steinfeld, and Jean Oh. Core challenges of social robot navigation: A survey. *ACM Transactions on Human-Robot Interaction*, 02 2023.
- [60] Rajesh Kannan Megalingam, Vignesh Naick, Sakthiprasad K.M, and Vinu Sivananthan. Analysis of tiago robot for autonomous navigation applications. pages 257–261, 08 2021.
- [61] Sariah Mghames, Luca Castri, Marc Hanheide, and Nicola Bellotto. A neuro-symbolic approach for enhanced human motion prediction, 2023.
- [62] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. *Layer-Wise Relevance Propagation: An Overview*, pages 193–209. Springer International Publishing, Cham, 2019.
- [63] Francisco-Angel Moreno, Javier Monroy, J.R. Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. Automatic waypoint generation to improve robot navigation through narrow spaces. *Sensors*, 20:240, 12 2019.
- [64] Vishnu K. Narayanan, Anne Spalanzani, François Pasteau, and Marie Babel. On equitably approaching and joining a group of interacting humans. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4071–4077, 2015.
- [65] Diego Paez Granados, Yujie He, David Gonon, Dan Jia, Bastian Leibe, Kenji Suzuki, and Aude Billard. Pedestrian-robot interactions on autonomous crowd navigation: Reactive control methods and evaluation metrics. pages 149–156, 10 2022.
- [66] Diego Paez Granados, Yujie He, David Gonon, Dan Jia, Bastian Leibe, Kenji Suzuki, and Aude Billard. Pedestrian-robot interactions on autonomous crowd navigation: Reactive control methods and evaluation metrics. 08 2022.
- [67] Jordi Pagès, Luca Marchionni, and Francesco Ferro. Tiago: the modular robot that adapts to different research needs. 2016.

- [68] Jong Jin Park, Collin Johnson, and Benjamin Kuipers. Robot navigation with model predictive equilibrium point control. pages 4945–4952, 10 2012.
- [69] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction, 2017.
- [70] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison W. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *CoRR*, abs/1704.02971, 2017.
- [71] U.U.Samantha Kumara Rajapaksha, Chandimal Jayawardena, and Bruce A. MacDonald. Ros based heterogeneous multiple robots control using high level user instructions. In *TENCON 2021 - 2021 IEEE Region 10 Conference (TENCON)*, pages 163–168, 2021.
- [72] People2D: realtime people detection in 2D range data.
- [73] Ely Repiso, Anaís Garrell, and A. Sanfeliu. Adaptive side-by-side social robot navigation to approach and interact with people. *International Journal of Social Robotics*, 12, 08 2020.
- [74] Ely Repiso, Anaís Garrell, and A. Sanfeliu. Adaptive side-by-side social robot navigation to approach and interact with people. *International Journal of Social Robotics*, 12, 08 2020.
- [75] Maria Ribeiro and Isabel Ribeiro. Kalman and extended kalman filters: Concept, derivation and properties. Technical report, 04 2004.
- [76] Meera Sebastian, Santosh Balajee Banisetty, and David Feil-Seifer. Socially-aware navigation planner using models of human-human interaction. pages 405–410, 08 2017.
- [77] Seif Eddine Seghiri, Noura Mansouri, and Ahmed Chemori. Implementation of sarl* algorithm for a differential drive robot in a gazebo crowded simulation environment. In *2022 2nd International Conference on Advanced Electrical Engineering (ICAEE)*, pages 1–6, 2022.
- [78] Ishneet Sethi, Alka Trivedi, Pranav Singhal, Mrinal Bhave, Rishika Agarwal, Rahul Kala, and Gora Chand Nandi. Group-aware human trajectory prediction. In *2022 IEEE 6th Conference on Information and Communication Technology (CICT)*, pages 1–5, 2022.

- [79] Phani Singamaneni, Anthony Favier, and Rachid Alami. Human-aware navigation planner for diverse human-robot contexts, 2021.
- [80] Kiran Jot Singh, Divneet Kapoor, and Balwinder Sohi. Understanding socially aware robot navigation. *Journal of Engineering Research*, 9, 10 2021.
- [81] Micol Spitale and Franca Garzotto. Socially assistive robots in smart homes: Design factors that influence the user perception. pages 1075–1079, 03 2022.
- [82] Yuxiang Sun, Ming Liu, and Max Q.-H. Meng. Active perception for foreground segmentation: An rgb-d data-based background modeling method. *IEEE Transactions on Automation Science and Engineering*, 16(4):1596–1609, 2019.
- [83] Phani Teja S. and Rachid Alami. Hateb-2: Reactive planning and decision making in human-robot co-navigation. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 179–186, 2020.
- [84] Xuan-Tung Truong and Trung Dung Ngo. Dynamic social zone based mobile robot navigation for human comfortable safety in social environments. *International Journal of Social Robotics*, 8, 11 2016.
- [85] Xuan-Tung Truong and Trung Dung Ngo. “to approach humans?”: A unified framework for approaching pose prediction and socially aware robot navigation. *IEEE Transactions on Cognitive and Developmental Systems*, PP:1–1, 09 2017.
- [86] Xuan-Tung Truong, Voo Nyuk Yoong, and Trung-Dung Ngo. Rgb-d and laser data fusion-based human detection and tracking for socially aware robot navigation framework. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 608–613, 2015.
- [87] Manav Vallecha and Rahul Kala. Group and socially aware multi-agent reinforcement learning. In *2022 30th Mediterranean Conference on Control and Automation (MED)*, pages 73–78, 2022.
- [88] Anirudh Vemula, Katharina Muelling, and Jean Oh. Modeling cooperative navigation in dense human crowds. pages 1685–1692, 05 2017.
- [89] Tianxi Wang, Feng Xue, Yu Zhou, and Anlong Ming. Marf: Multiscale adaptive-switch random forest for leg detection with 2d laser scanners, 2022.

-
- [90] Pan Wei, Lucas Cagle, Tasmia Reza, John Ball, and James Gafford. Lidar and camera detection fusion in a real-time industrial multi-sensor collision avoidance system. *Electronics*, 7(6), 2018.
- [91] Shunyi Yao, Guangda Chen, Quecheng Qiu, Jun Ma, Xiaoping Chen, and Jianmin Ji. Crowd-aware robot navigation for pedestrians with multiple collision avoidance strategies via map-based deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8144–8150, 2021.
- [92] Leon Jung Yoonseok Pyo, Hancheol Cho and Darby Lim. *ROS Robot Programming (English)*. ROBOTIS, 12 2017.
- [93] Francesco Zanlungo, Claudio Feliciani, Zeynep Yücel, Katsuhiro Nishinari, and Takayuki Kanda. Macroscopic and microscopic dynamics of a pedestrian cross-flow: Part i, experimental analysis. *Safety Science*, 158:105953, 02 2023.
- [94] Francesco Zanlungo, Claudio Feliciani, Zeynep Yücel, Katsuhiro Nishinari, and Takayuki Kanda. Macroscopic and microscopic dynamics of a pedestrian cross-flow: Part ii, modelling. *Safety Science*, 158:105969, 02 2023.

Acknowledgments

I would like to express my heartfelt gratitude to everyone who has contributed to the completion of my academic path and my master's thesis in Computer Engineering at the University of Padova. It has been a long and challenging journey, but it would not have been possible without the support and encouragement of so many wonderful people.

First and foremost, I would like to thank my parents for the unconditional love, unwavering support, and endless encouragement they made me feel every day. Their sacrifices and dedication have been instrumental in helping me pursue my dreams and achieve my goals. I cannot thank them enough.

I would also like to express my gratitude to my brother. His huge belief in me has pushed me to strive for excellence, and I am grateful for his love and support.

I would like to acknowledge the support I received from my professors, who have been instrumental in shaping my academic and professional growth. Their guidance, constructive feedback, and insightful discussions have broadened my horizons and helped me develop my research skills.

I would also like to thank my friends for their continuous encouragement, which has kept me motivated and energized throughout my studies both in the positive and in the negative moments we spent together. In particular, I have to express a special appreciation to my friends who took part in my experiments.

I would like to convey my sincere gratitude to my trainer, whose guidance, expertise, and mentorship have played a daily support and crucial role in shaping my personality and my ability to approach challenges with a resilient and victorious mindset.

Finally, I would like to greet the contributions of all the people who helped me to review my thesis since they provided me valuable feedback and suggestions to improve the quality of my work.

Thank you all. I will always cherish the memories of this journey.