



SAPIENZA
UNIVERSITÀ DI ROMA

Enhancing robot reliability for health-care facilities by means of Human-Aware Navigation Planning

Faculty of Information Engineering, Informatics, and Statistics
Master Course on Artificial Intelligence and Robotics

Olga Sorokoletova

ID number 1937430

Advisors

Prof. Christian Napoli

Prof. Luca Iocchi

Dr. Samuele Russo

Academic Year 2021/2022

Thesis defended on 31 January 2023
in front of a Board of Examiners composed by:
Prof. Aristidis Anagnostopoulos (chairman)
Prof. Danilo Comminiello
Prof. Giorgio Grisetti
Prof. Christian Napoli
Prof. Fabrizio Silvestri
Prof. Andrea Vitaletti

**Enhancing robot reliability for health-care facilities by means of Human-Aware
Navigation Planning**
Sapienza University of Rome

© 2023 Olga Sorokoletova. All rights reserved

This thesis has been typeset by L^AT_EX and the Sapthesis class.

Author's email: sorokoletova.1937430@studenti.uniroma1.it

Abstract

With the aim of enabling robots to cooperate with humans, carry-out human-like tasks or navigate among humans we need to ensure that they are equipped with the ability to comprehend human behaviours and use the extracted knowledge for the intelligent decision-making. This ability is particularly important in the safety-critical and human-centered environment of health-care institutions where, on the one hand, proactive, contextualized, thoughtfully designed robots could be beneficial to free up providers' time from low-value tasks and allow them to focus more on patient care, but, on the other hand, errors could endanger patients' lives.

This work is centered on the navigation branch of the robotic research. On that field the most cutting-edge approaches to enhance robot reliability in the application domain of health-care facilities and in general, pertain to augmenting navigation system with human-aware properties, or, in other words, to establishing the Social Navigation. Therefore, the aim was to incorporate the human-aware planner into the navigation module of the robot architecture.

As a result, **Co-operative Human-Aware Navigation planner**, designed by our collaborators from the LAAS-CNRS¹ research group, **has been integrated** into ROS-based differential-drive robot MARRtino² **and exhaustively challenged within various simulated contexts** and scenarios (mainly, modelling the situations, relevant in medical domain) to draw attention to the integrated system's benefits and identify its drawbacks or instances of poor performance while exploring the scope of system capabilities and creating a full characterisation of its applicability. The simulation results are then presented to medical experts, and **the enhanced robot acceptability within domain is validated** with them, as the robot is further planned for deployment.

¹LAAS-CNRS, Universite de Toulouse, CNRS, Toulouse, France, {ptsingaman, anthony.favier, rachid.alami}@laas.fr

²<https://www.marrrtino.org/home>

Acknowledgments

I would like to express my gratitude to everyone who had been standing by my side with their expertise or care while I was pursuing my passions in the Master's course of Artificial Intelligence and Robotics and, especially, while I was developing this thesis that I found challenging, but at the same time very exciting and inspiring for future investigations of the area of Social Navigation.

First, I would like to thank people who gave me the opportunity to develop my thesis. Thank you, professor Christian Napoli, for your supervision. Thank you, Phani-Teja Singamaneni and Rachid Alami – our dearest colleagues from LAAS-CNRS research group who developed the planning approach that I have been integrating, made it accessible as an open source project and agreed to support me in my work as well as professor Luca Iocchi who made this collaboration possible by introducing us to each other and patiently guided me throughout the process. Thank you for being always available, responsive, kind, ready to provide expert opinions and opened to questions and discussions of new ideas. Then also very important, thank you, doctor Samuele Russo and all the colleagues from Sant'Andrea Hospital who validated the results of the system performance and helped us evaluate the human acceptability from perspective of specialists of application domain. I am genuinely happy that I had a chance to work in this big collaboration with all the people involved and I hope that in the future the project will overgrow itself into something even better.

Then, I would also like to thank my family and especially my mom for never-ending love, believing in me, understanding and supporting me in my decision to study abroad.

Next, I want to thank my old friends, Ekaterina, Maria, Natalia, Evgeny, Alexander and many others, for being eager to follow my progresses and making me feel that no matter the distance you are always there and my new friends, Amila, Nadina, Alena, Edna, Lorenzo, Leandro, Giorgia, Adriano, Antonio, Lorenzo, Leonardo and

again many others, for making these two years rich of enjoyable moments and memorable experiences. Thank you, Ekaterina, for following each and every of my steps. Thank you, Amila, for being my best friend in Rome. This time now associates with you. Thank you, Lorenzo, Leandro and Giorgia, for the best experience of the project work I had. Thank you all for supporting me in stress and tough moments. Separate thank you to my Italian friends for introducing me into their culture and explaining the administrative aspects. It was a pleasure to share my time with you and it would be even more pleasure to preserve our friendship over years.

Last but not least, I would like to say how much I appreciate my couple Daniele for staying by my side always, in good and bad moments, being patient during these months and trying to understand me and find a way to help me even when it was impossible.

Contents

1	Introduction	1
1.1	Problem Definition	2
1.2	Overview on used Techniques	3
1.3	Thesis Contribution	5
1.4	Outline	5
2	Background and Related Works	7
2.1	Robotic Operation System	7
2.1.1	ROS Ecosystem	7
2.1.2	ROS Navigation Stack	11
2.2	Timed Elastic Band	14
2.2.1	Optimisation Problem	15
2.2.2	Extension	16
2.3	Human-Aware Navigation	18
2.3.1	Features of Human-Aware Navigation	19
2.3.2	Challenges of Human-Aware Navigation	19
2.4	Related Works	22
3	Co-operative Human-Aware Navigation Planner	25
3.1	Human Tracking	27
3.2	Human Path Prediction	29
3.3	Planning Modes	30

3.4	Human-Aware Constraints	34
4	Implementation	35
4.1	MARRtino software	35
4.2	Planner Integration	37
4.2.1	Stage Container	37
4.2.2	Navigation Container	38
5	Experiments	41
5.1	Experimental Setup	41
5.2	Results	43
5.2.1	Qualitative Results	44
5.2.2	Quantitative Results	68
5.2.3	Human Evaluation Results	70
6	Conclusion and Future Work	75
	Appendix	79
A	Final System Configuration	79
	Bibliography	85

Chapter 1

Introduction

Taking into account the specificity of the application domain is one of the crucial considerations while developing a robotic system, and the chosen domain of health-care facilities is represented by the quick-paced and safety-critical environment which is frequently overcrowded, chaotic and understaffed. The health-care workers (HCWs) who must operate in this setting on a daily basis are often overburdened and run the risk of experiencing the mental exhaustion which is not only harmful for them, but also dangerous for the patients since mental exhaustion of providers makes them more prone to mistakes. Additionally, the patients' perception of a poorer level of treatment affects them psychologically which has a substantial impact on the speed of recovery. Therefore, the robotics community has been researching the use of robots in hospital settings to reduce the workload of caregivers by utilising robots for low-value duties such as medical supplies delivery or patients assistance at the bedside when clinicians are not available. The thesis goal is to contribute to the aforementioned area of research by increasing reliability of the autonomous mobile robots, deployed to spare a large amount of time that is generally wasted by the human providers and decrease their burden in Intensive Care Units (ICU), Emergency Departments (ED) and health-care wards in general.

1.1 Problem Definition

Nowadays, the number of autonomous mobile robots, serving in hospitals, is rather low, if not extremely low when we confine ourselves to robots as fully operational and independent units. This reveals a major gap in terms of existing approaches and is explained by the fact that the higher the desired level of autonomy is, the more challenging is to provide it while also satisfying safety criteria of a primary importance for the applications of this work. Thus, the clinical environment is a unique **safety-critical environment** that poses specific navigation challenges for robots [40], [41] and gives rise to particular scenarios in which the navigational conflicts between humans and robots must be resolved.

For example, a robot in ICU is likely to encounter the situation when a group of HCWs, some of whom may have been involved in other tasks and the predictions of expected behaviours for whom may have been already built by the robot’s planner, would suddenly rush to the bed of a certain high-acuity patient (e.g. such as having a heart attack or stroke) to perform a life-saving treatment. If the robot’s objective is next to the area where this high-acuity patient is located, the robot keeps pursuing its objective instead of abandoning it and, as a result of not being lucky enough with timings, interrupts a clinical team, it could lead to a fatal outcome for the patient. Such robot would not be desired for placement in ICU. To address the challenge of performing the situation assessment and responding to changes in dynamics of the environment due to the changing human plans, robots must be more than purely reactive – they must be **proactive** and **adaptive**, implying, in turn, two things: 1) being **anticipative**: robots must be able to understand the behaviour and anticipate the intentions of the people, working around them in the context of health-care environments, 2) being capable of **intelligent decision-making**: the extracted knowledge and generated predictions must be taken into consideration in the intelligent, human-aware navigation decisions made by robots.

Besides, there is another kind of **adaptability**, required from the robot – to

a new environment with different characteristics. For example, let us assume that in a particular hospital ICU is a room in the ED floor. The robot's initial position is in ICU and it is tasked to exit and proceed to its goal, navigating through ED. The ED often has patients placed in the hallways, making it difficult to go around without disturbing anyone. Moreover, ICU could resemble a crowded open space kind of environment (beds are normally installed along the walls, however, there could be a wall and/or nursing station in the centre of the room), meanwhile, the corridors of ED could be not so crowded, but cluttered with trolleys and appliance. Additionally, providers may need to quickly move patients through these corridors. When a patient with a high level of acuity needs to be moved through a corridor that the robot is navigating and the corridor is only wide enough for one agent, the robot must immediately backtrack in a proactive manner. In the ICU, the scenario wasn't as relevant.

1.2 Overview on used Techniques

In this thesis, the **Co-operative Human-Aware Navigation (CoHAN) planner** [37] which enables a robot to navigate in diverse contexts while being aware of humans in the environment is integrated to address the described challenges and mitigate inconsistencies between the robot's actual and expected by people behaviour. We use a **ROS-based differential drive MARRtino robot** with functionalities, equivalent to a Turtlebot system in [Figure 1.1a](#), with the intention of deploying its social version (also known as MARRtina in [Figure 1.1b](#)) to serve as a bedside robot, aiding patients/HCWs in Sant'Andrea Hospital in Rome. As a result of integration, when MARRtino produces paths, it performs a **Social Navigation** as opposed to treating people simply as obstacles, as all the previously integrated planners did.

The Social (or Human-Aware) Navigation is a relatively new branch of robotic navigation that states the robots' paths must not only be safe, legible and optimal in

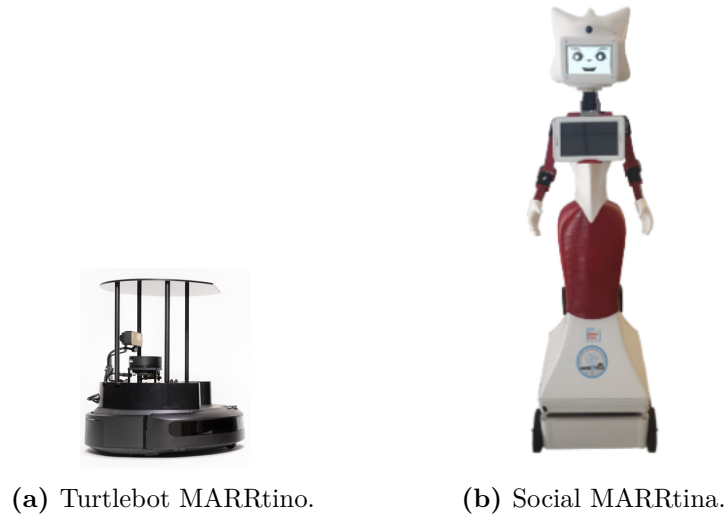


Figure 1.1. MARRtino robot versions available.

terms of time and robot resources consumption, but also acceptable/looking natural to humans. Then, since Human-Aware Navigation focuses on developing more human acceptable robotic systems, it needs to investigate and account for various human motion patterns and social aspects, relevant in navigation. The associated research community has explored a wide range of interesting subject matters so far and developed a variety of Human-Aware Navigation planners based on them, however, these planners are mostly environment/scenario-specific and hence do not meet the adaptability requirement, indicated above. Therefore, the motivation behind choosing for integration the CoHAN planner is that it is a flexible, highly tunable and easily adjustable to the various contexts system which is capable of handling complex indoors and crowded scenarios (ICU, ED). Moreover, refining and upgrading of a CoHAN system is an ongoing research. The main accent in a newly integrated planner is done on improvement of the **Timed Elastic Band (TEB) ROS Local Planner** [5] to make it **Human-Aware (HATEB)** [36] by incorporating **human motion predictions and estimations and simultaneous planning** for humans and the robot. The system supports multiple human path prediction services as well as multiple modes of planning with an intelligent mechanism of switching between them.

1.3 Thesis Contribution

The main contribution of this work is that in addition to being **integrated** into MARRtino software architecture (which is different from the original PR2 robot architecture that was used for development), a novel state-of-the-art Human-Aware Navigation planner has been **thoroughly challenged, tested and evaluated in numerous simulated human-robot scenarios and environments**: ICU crowded free space, ICU with a wall in the middle, heart attack emergency in ICU, narrow hallway, free/cluttered wide ED corridor, narrow/wide door crossing scenario, patient bed approach scenario, etc. The thesis objective is to present both qualitative and quantitative analysis while uncovering subtle nuances of the system's functioning and ultimately creating a **comprehensive characterisation of the properties and limitations of the adopted planning approach**. As such, globally, the goal is to contribute to better patient outcomes and lessen the workload of HCWs by making robots more reliable and moving us closer to the time when robots will be able to autonomously socially navigate in a real safety-critical health-care environment (as well as in other applications) without endangering or confounding humans around. One of the necessary steps towards this goal is to ensure that potential users accept the developed robotic technology. Therefore, after gathering feedback on the system performance from the HCWs at the Sant'Andrea Hospital, a brief **statistical analysis of the robot's acceptability** by humans was carried out.

1.4 Outline

The rest of the document is structured as follows. For a better understanding of the subsequent chapters, the background material is presented in [Chapter 2](#). It discusses reasons for choosing ROS and explains ROS 2D Navigation Stack, in general, and TEB Local Planner, in particular. Then, the Human-Aware Navigation Planning problem is described, and some of the related works are commented. The

CoHAN planner’s architecture, including overview of its features, components and the scheme of communication between different modules and processes, is provided in [Chapter 3](#). Following this, [Chapter 4](#) deals with the details of integration of a CoHAN system into MARRtino software. The primary matter of the work is found in [Chapter 5](#). It covers a description of the experimental setup, a qualitative and quantitative analysis of the integrated system’s performance and a statistical analysis of its acceptability by a group of possible future users. Finally, the thesis is summarised in [Chapter 6](#) which also comments on potential future research to improve the system.

Chapter 2

Background and Related Works

This chapter explains in detail the fundamentals of the framework, method and research area involved: the Robotic Operation System (ROS) software for developing robot applications with its 2D Navigation Stack collection of packages, Timed Elastic Band (TEB) local trajectory planner and the field of Human-Aware (Social) Navigation. At the end, it also provides an overview on the related works.

2.1 Robotic Operation System

Both MARRtino robot software and Co-operative Human-Aware Navigation (CoHAN) planner that we integrate into it are ROS-based. Therefore, it seems reasonable to begin by introducing and motivating the use of ROS Ecosystem, consequentially narrowing the focus to the ROS Navigation Stack collection of software packages.

2.1.1 ROS Ecosystem

ROS is an **open-source robotic middleware toolset** that provides a structured communications layer above the host operating systems [32].

History and Motivation

In the early days of robot software development, before ROS and analogical robotic middlewares existed, the architectures of the robot applications mainly consisted of monolithic modules, meaning that logically distinct functionalities could have been encapsulated in the same programming module, like, for instance, action-related and perception-related functionalities could have been mixed. This, in turn, resulted in a gigantic convoluted dependency trees representing the architecture of the software under development. Furthermore, such systems were unreliable and prone to failures because a single module crash could cause collateral damage to an entire system.

As a solution, robotics researchers directed their attention to distributed systems, where different functional components are distributed across different processes that communicate and coordinate their actions by passing messages to one another. Advantageously, **the distributed systems are far more reliable and scalable** because a single process malfunctioning does not impose problems for the remaining processes and whenever there is a need, a single component can be updated/upgraded/replaced independently on others as well as new components can be added to the system with minimal effort. Since then, a wide range of frameworks for robotic software development and advancement with a distributed, modular design has been created by the robotic community. Many of these frameworks are in use in research and industry, and they are capable of managing complexity and integrity of the systems being developed.

ROS was designed as part of the Stanford Artificial Intelligence Robot (STAIR) project at Stanford University [4] and the Personal Robots Program [45] at Willow Garage. As it is emphasized in [32], it would be unfair to say that ROS is the best set of the frameworks for any robotic application because such a universal solution, probably, cannot exist since the field of robotics is far too broad for a one-fit-all solution to exist. However, it may be true that **ROS is the most general-purpose**

and commonly-used one because of the number of reasons:

- Collaborative community that shares software and code and helps in maintenance;
- Peer-to-peer connection topology;
- Scalability;
- Tools-based approach;
- Multi-linguality (C++, Python, Octave, LISP, MATLAB, etc);
- Thin ideology (code is reusable);
- Free and open-source;
- Advanced easy-to-use simulation and testing utilities.

For example, in the context of this thesis the MARRtino robot's ROS-based software was given and the goal was to improve a particular component – Navigation Planner. The software already had other planners included, but none of them was human-aware. Since the CoHAN planner was also written in ROS, it was compatible for integration into Navigation Stack and testing with Stage and RViz tools that are normally used to test MARRtino. Moreover, thanks to flexibility of ROS, the new architecture allows the instant switching from CoHAN to any previously integrated planner or re-integration of a newer version of CoHAN.

System Organisation

When it comes to ROS implementation, in a nutshell, there are three levels of concepts: the Community level, the Computation Graph level, and the Filesystem level [1].

1. **The Community level** concepts are ROS resources that allow for a knowledge and code sharing and exchange between developers.

2. **The Computation Graph level** is a peer-to-peer network of ROS family of processes. The basic communication scheme is shown in [Figure 2.1](#), and it depicts four fundamental concepts of the level:

- **Nodes** – the processes that perform computation;
- **Messages** – means of communication between the nodes (nodes communicate by passing messages to each other);
- **Topics** – a node sends a message by publishing it to a given topic;
- **Services** – implementation of synchronous request/reply transactions.

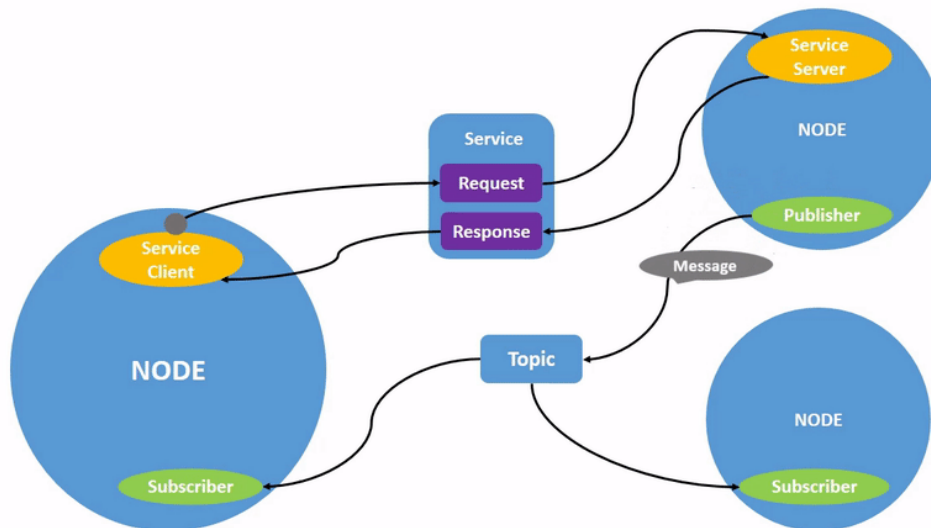


Figure 2.1. Basic communication between ROS nodes: topic-based publish-subscribe and service-based request-response model.

Besides them, the Graph level covers three more concepts:

- **Master** – the core that acts as a nameservice and stores topics and services registration information for ROS nodes;
- **Parameter Server** – allows data to be stored by key in a central location;
- **Bags** – mechanism for storing message data.

3. **The Filesystem level** describes ROS meta-operating system properties since it provides some operating system-like functions, e.g. hardware abstraction.

Figure 2.2 illustrates organisation of ROS files on a hard drive.

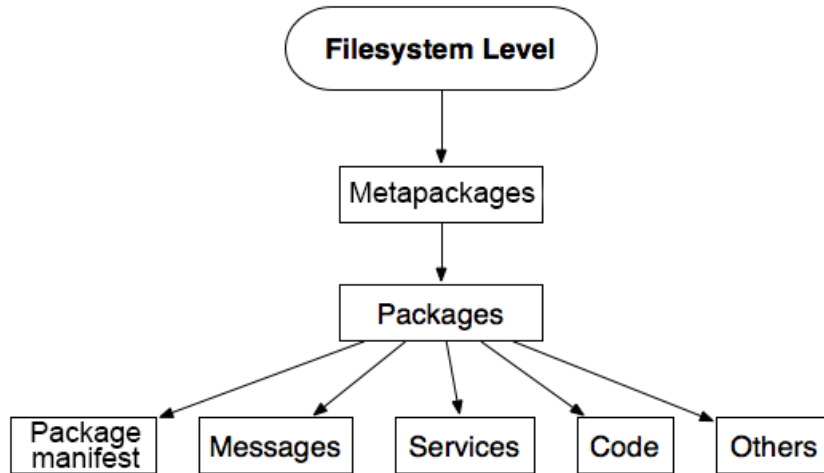


Figure 2.2. ROS Filesystem level.

As it can be seen, the main concepts of the level are packages, metapackages, package manifests, etc. Packages may contain source codes, programs (nodes), libraries, configuration files and datasets. They are the atomic units for organising software in ROS, meanwhile the other concepts of the level – workarounds to provide the package management. One of them is **ROS Stack – the collection of packages**. In particular, to navigate with a ROS robot, the ROS Navigation Stack, described below, is used.

2.1.2 ROS Navigation Stack

The ROS navigation [2], or ROS Navigation Stack, is meant for 2D maps. It takes as input an information about odometry (e.g. from wheel encoders, IMU, GPS), sensor measurements (e.g. of laser) and a goal pose to provide as output safe velocity commands that are sent to a mobile base.

Functionally, the Navigation Stack is a set of ROS nodes and navigation-related algorithms that are implemented to move mobile robot autonomously from one

point to another while avoiding obstacles. It contains packages implemented to create maps of environments, localise the robot in the environment, perform Path Planning, visualise data of navigation processes, configure different navigation nodes and debug errors. The ROS Navigation Stack's basic structure is represented by the three main blocks/modules:

- **Mapping** – packages to create a map of the environment;
- **Localisation** – packages to localise a robot in the created map;
- **Path Planning** – packages to plan the robot's path (**Global Path Planning**) and execute the planned path while avoiding obstacles (**Local Path Planning**).

The following subsections explain each of the main blocks in more detail. However, let us note that there exist also techniques, approaches and concepts that lie at the intersection of different blocks: Simultaneous Localisation and Mapping (SLAM) (Mapping + Localisation), Active Localisation (Localisation + Path Planning), Exploration (Path Planning + Mapping), etc.

Mapping

The navigation in ROS always starts with a Mapping – **the process of building a map** – a representation of an environment created from the sensor measurements of the robot (e.g. let us consider laser). For that robot needs to move in the environment and acquire the laser scans. In ROS nomenclature it means executing the launch file that is a map builder node. Then, if Mapping is done in a simulation, the teleoperation can be launched to manually move a robot in order to make it explore the environment.

Localisation

The next step after Mapping is to provide a robot with knowledge about its **position and orientation in the built map** (in the reference frame of a metric

map) at every timestep. This process in ROS is called Localisation. As the Localisation is launched, the localisation algorithm produces the number (can be configured) of guesses. Then, the robot again can be moved using teleoperation, and the most likely (usually not perfect) guess/estimate of the robot pose is the result of the process. Otherwise, the estimation can be provided to the algorithm through RViz tool.

Path Planning

Finally, the Path Planning module takes as input a localised robot, a map and a desired position and orientation of the robot (navigation goal) to give as **output the best (according to some optimisation criteria, e.g. shortest) path** which is a sequence of waypoints or actions to reach the navigation goal, if such a sequence exists.

Taking into consideration the entire map with known in advance from Mapping and never changing distribution of free/occupied locations in the map, the global path is produced by the Global Planner component. Using this component exclusively works well in environments that never change and do not contain dynamic obstacles (hence without people) and when sensor readings are perfect. Unfortunately, nothing of the aforementioned is a property of the most real-world environments. Thus, the complementary design component is required to enable for the obstacle avoidance and plan refinement – the Local Planner.

In a sense, **the problem of Path Planning can be split into two sub-problems: how to generate a path (Global Planner) and how to follow it (Local Planner)**. As a part of path following, the Local Planner breaks down the map into smaller areas, and updates the global path by modifying it locally using the real-time sensor measurements. It produces a local path – the best (according to the local optimisation problem criteria) path for the area it currently traverses.

The basic navigation pipeline in ROS would require executing the `move_base`

node and sending the 2D Navigation Goal using RViz. The `move_base` in [Figure 2.3](#) is the main node in ROS Navigation Stack whose function is to move a robot from its current position to a goal position with the help of other navigation nodes. This node links: 1) Global Planner and Local Planner for the Path Planning, 2) Rotate Recovery package for the use when the robot is stuck, 3) Global Costmap and Local Costmap, containing the information about the obstacles.

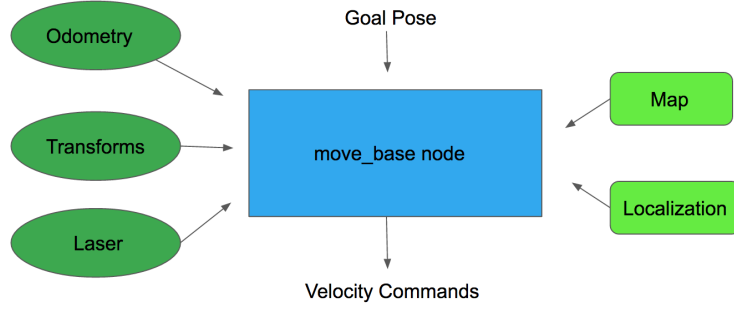


Figure 2.3. Architecture of the `move_base` node’s inputs-outputs.

2.2 Timed Elastic Band

The component of just described ROS Navigation Stack that was modified to provide the social awareness during navigation is a Local Planner. Precisely, Timed Elastic Band optimal local trajectory planner was taken as a foundation and then expanded with human-aware characteristics. It is implemented in ROS in a `teb_elastic_band` package as a plugin to the `base_local_planner` – default Local Planner of the 2D Navigation Stack. The current section is devoted to the detailed description of the underlying planning approach that was introduced in [\[34\]](#) and whose core idea is in optimising locally at runtime the initial trajectory generated by a Global Planner with respect to:

- **Trajectory execution time** (time-optimal objective);
- **Separation from the obstacles**;

- **Kinodynamic constraints** (maximum velocities and accelerations).

First, the elastic band is a sequence of n intermediate robot configurations $\mathbf{x}_i = (x_i, y_i, \beta_i)^T \in R^2 \times S^1$, where (x_i, y_i) is a position and β_i – orientation of the robot in a map reference frame:

$$Q = \{\mathbf{x}_i\}_{i=0\dots n} \quad n \in \mathbb{N} \quad (2.1)$$

Then, the Timed Elastic Band (TEB) appears as an augmentation by the time intervals between two consecutive configurations: $n - 1$ time differences ΔT_i each of which is the time, needed to transit from one configuration to another, as represented in [Figure 2.4](#).

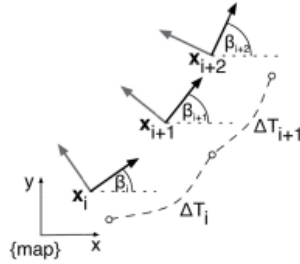


Figure 2.4. Structure of TEB: sequence (band) of configurations and time differences.

Therefore, **the TEB is a tuple of sequences**:

$$B := (Q, \tau) \quad \tau = \{\Delta T_i\}_{i=0\dots n-1} \quad (2.2)$$

2.2.1 Optimisation Problem

The Optimisation Problem can be formulated as a **real-time weighted multi-objective optimisation of TEB** defined in [Equation 2.2](#) in terms of both configurations and time intervals. Let us denote $f(B)$ – objective function and B^* –

optimal values (problem solution), then:

$$f(B) = \sum_k \gamma_k f_k(B) \quad (2.3)$$

$$B^* = \arg \min_B f(B) \quad (2.4)$$

where γ_k are the weights and f_k – multiple objectives that may capture the diverse aspects, but mainly can be divided into two groups:

- Constraints that are formulated as objectives in terms of a piecewise continuous differentiable cost function that penalizes the violation of a constraint:

$$e_\Gamma(x, x_r, \epsilon, S, n) \simeq \begin{cases} \left(\frac{x - (x_r - \epsilon)}{S}\right)^n & \text{if } x > x_r - \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (2.5)$$

where x_r , S , n and ϵ are the bound, scaling factor, polynomial order and a small translation of approximation, correspondingly;

- Objectives with respect to trajectory (e.g. shortest path).

The most of the objectives are local with respect to B because they normally depend on only few consecutive configurations rather than the entire sequence. Consequentially, the system matrix is sparse and optimisation problem can be solved using specialised numerical methods if they are implemented in a used framework.

2.2.2 Extension

The ROS `teb_elastic_band` package contains an extended version of TEB. **Extension consists in planning of multiple robot trajectories in Distinctive Topologies** so that a subset of admissible trajectories of distinctive topologies is optimised in parallel [35]. By exploiting this method, the Local Planner is able to switch to the current globally optimal trajectory among the candidate set. Distinctive Topologies, in turn, are obtained by utilising the concept of Homology/ Homotopy Classes:

- **Homotopy Class** is a set of all trajectories that are homotopic to each other. Two trajectories τ_1 and τ_2 connecting the same start and goal positions z_s and z_g respectively, are homotopic if and only if one can be continuously deformed into the other without intersecting any obstacles;
- **Homology Class** is a set of all trajectories that are homologous to each other. Two trajectories τ_1 and τ_2 connecting the same start and goal positions z_s and z_g , respectively, are homologous if and only if $\tau_1 \sqcup -\tau_2$ forms the complete boundary of a 2D manifold, embedded in \mathbb{R}^2 . In simpler words, it means going from start to goal with τ_1 and returning with τ_2 without surrounding any obstacle.

Figure 2.5 illustrates the Homotopy and Homology in 2D. In this example, τ_1 and τ_2 are both homotopic (because of the existence of the sequence of trajectories shown by the dashed curves) as well as homologous (because of the area shown by blue hashing). But τ_3 is not homotopic nor homologous to either. Therefore, there are two Homology and Homotopy Classes: one is formed by τ_1 and τ_2 , and another one – by τ_3 .

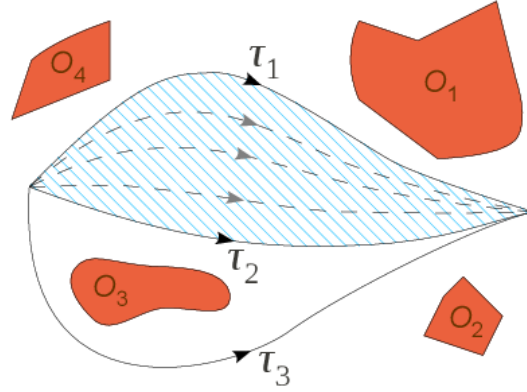


Figure 2.5. Illustration of Homotopy and Homology Classes in 2D, reproduced from [9].

In this example τ_1 and τ_2 are both homotopic as well as homologous. But τ_3 is not homotopic nor homologous to either.

The closed form generic computation of Homotopy Classes is difficult. The suggested approach to avoid this difficulty is in substituting the Homotopy with Homology Classes as they are easier to compute. In example in Figure 2.5 Homology

Classes are coincident with the Homotopy Classes. In general, this is not the case: homotopy implies homology, but the reverse implication does not hold. However, in the most planning scenarios homotopy and homology can be considered equivalent.

The default setting of the elastic band in TEB allow it to rapidly change the Homotopy Class. This works well when the robot is far from humans. However, in the close vicinity of humans it may cause the loss of legibility of the robot’s motion. This issue is addressed in a modified TEB, HATEB [36], by restricting the change of Homotopy Class under a threshold distance (tunable parameter) from humans. **The restriction is implemented by decreasing the time interval between consecutive poses which tightens the elastic band.**

In this thesis, we focus on another extension of TEB – an extension that consists in encapsulating the features of deriving socially compliant paths. With such features TEB becomes a Human-Aware TEB (HATEB) – the algorithm of the Social (Human-Aware) Navigation area, described in the following section.

2.3 Human-Aware Navigation

In the last years a multidisciplinary field of Human-Robot Interaction (HRI) that studies the interaction dynamics between humans and robots has been attracting a lot of attention. The **Human-Aware Navigation** is another field that emerged relatively recently from the **synthesis of HRI and Motion Planning** domains. A crucial problem, coming for the nature of the synthesis, is the problem of balancing different objectives in such a way that implemented accounting for social aspects does not take precedence over efficiency (time, energy, etc.) criteria. This makes it challenging to provide features and meet challenges of Human-Aware Navigation, described in more detail below.

2.3.1 Features of Human-Aware Navigation

The following features are conventionally included into Human-Aware Navigation frameworks, as [25] suggests:

- Respecting the personal zones and affordance spaces;
- Avoiding of culturally unacceptable behaviours;
- Avoiding of erratic motions and distracting noises;
- Reducing the velocity when approaching human;
- Approaching from the front for explicit interaction;
- Modulating gaze direction;
- Emitting non-verbal communication signals other than gaze;
- Recognizing and reacting to non-verbal communication signals.

Each of the features is significant in addressing one (or more) of the Human-Aware Navigation challenges which are discussed in the next subsection.

2.3.2 Challenges of Human-Aware Navigation

In general, the robot navigation in presence of humans has always been a challenge that can be defined as a **challenge of navigating while accounting for the constraints related to social aspects/rules** (e.g. human comfort) [25]. More precisely, this is not just one, but the group of challenges that address the common objective of enhancing robot acceptability by humans.

One of the possible approaches to sub-categorization of characteristics that the robot must exhibit in order to be considered navigating in a human-aware manner (raising the corresponding challenges of providing these characteristics) concentrates attention on comfort, naturalness of motion and sociability:

- **Human Comfort** is defined as absence of annoyance and stress for humans during human-robot interaction session or in shared spaces;
- **Motion Naturalness** is the ability of the robot to elaborate human-alike behaviour by capturing and recreating human low-level behaviour patterns in navigation;
- **Sociability** is the ability to elicit high-level cultural conventions and social norms.

Human Comfort

Human comfort is linked to the concept of safety and even employs a similar high-level two-step tackling strategy: detect the risk/uncomfortable behaviours → eliminate it/change the behaviour. However, **the safety is concerned about real physical threats to the human, whereas the human comfort focuses on the perceived by humans level of threat**. The perceived level of threat may differ from the real one.

Good results have been achieved in the task of increasing human comfort in cases when the robot's behaviour is obviously uncomfortable. For instance, there have been implemented features of extra distances on top of safety distances to address the personal space violation issue, inspired by the study of **proxemics** – a branch of knowledge that deals with the amount of space that people feel it necessary to set between themselves and others, illustrated in [Figure 2.6](#). Another example is the ongoing work on **intelligent noise reduction** in vicinity of humans. However, when there is no obvious discomfort, but the level of human comfort can still be increased, robotic researchers have difficulty determining which motion patterns are less interfering.

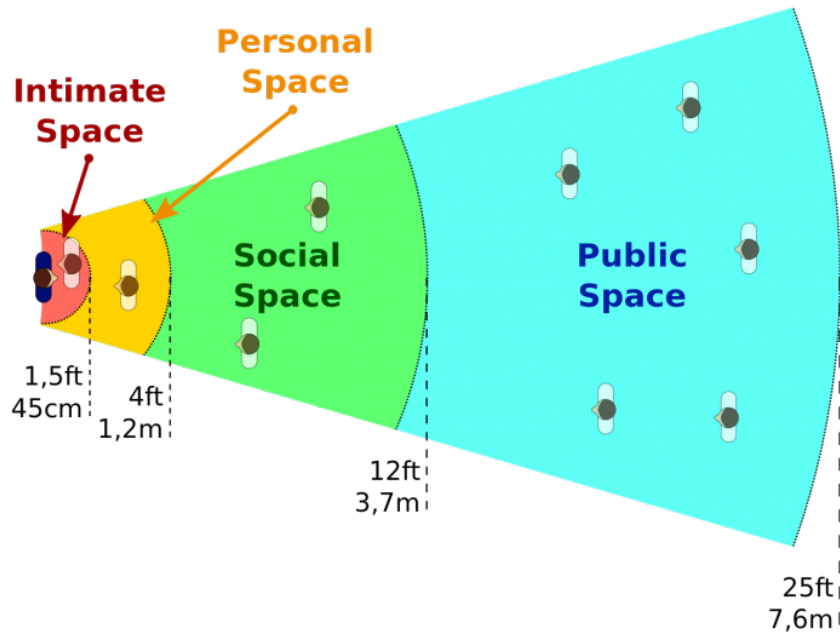


Figure 2.6. PROXIMITY. **Intimate distance** – indicates a closer relationship; **Personal distance** – occurs between family members or close friends; **Social distance** – used with individuals who are acquaintances; **Public distance** – used in public speaking situations.

Naturalness and Sociability

Both Motion Naturalness (the study of natural motion) and Sociability (the study of motion under social constraints) approach the problem of increasing robot acceptability by reducing differences between robot and human motions. However, they operate on the different levels.

The Motion Naturalness aims to mimic the nature of human motion and respond to the **low-level** question of whether the robot is capable of **reproducing the human motion patterns**. The main struggle in this research area resides in the fact that robots have different abilities compared to humans, and therefore not all patterns are meant to be transferred.

Sociability, on the other hand, is concerned with **high-level decision-making** and thus addresses a different question: whether a robot makes the same motion-related decision as a human. It attempts to comply with **social/cultural norms**

(e.g. prefer the right-hand side of the corridor, approach from the front, etc.) by either explicit modelling of known social protocols or learning them from humans as spacial effect. According to the recent research, learning from humans is especially advantageous because treating humans as intelligent social agents allows their behaviour to be not only predicted but also affected and exploited (e.g. for spacial conflicts resolution).

2.4 Related Works

The Co-operative Human-Aware Navigation (CoHAN) planner that has been integrated into framework was introduced by Singamaneni, Favier and Alami in [37]. This planner was designed to handle various human-robot contexts. Because **multi-context** Human-Aware Navigation Planning is still in its early stages, there have not been conducted many research activities on the subject. A method of layered costmaps, introduced in [26], a multi-objective optimisation approach, presented in [7], and a deep learning approach, described by the authors of [8], can be considered as some of them. In general, there exists a variety of alternative to CoHAN Human-Aware Navigation planners, however, the majority of them has been developed and tuned for a specific context/type of environment.

Concentrating first on **crowded environments**, the idea of the so-called Social Force Model (SFM) was introduced by the authors of [14] and complemented by Truong and Ngo [43] and Repiso et. al [33] who proposed an effective Proactive Social Motion Model (PSMM) and the Extended Social Force Model (ESFM) for a robot-companion, respectively. The works [12], [42] and [28] approach the same problem with Reinforcement Learning (RL) and Inverse Reinforcement Learning (IRL) instead.

Shifting to a Human-Aware Navigation in **indoor contexts**, the groups of researchers in [39], [44] and [22] contributed into development of the costmap-based approaches, as the one employed in CoHAN to deal with static humans. A relatively

recent case study for a warehouse environment in [11] discusses inclusion of an HRI model in the planning process. Then again, similarly to a crowded scenario, applications of both RL [18] and IRL [30], [29] families of approaches have been explored. In particular, Taylor et. al [41] introduced the Safety-Critical Deep Q-Network (SafeDQN) RL model for navigation with mobile robots in Emergency Department. Finally, for confined spaces, a planner, capable of proactively proposing co-navigation solutions in human-robot contexts and replicating human-like navigation behaviour, was presented in [21].

In its turn, proactive, interactive, human-like navigation behaviour in CoHAN planner is provided for a variety of contexts. This is mainly done by offering two types of flexibility: multi-modality and multiple Human Path Prediction services.

The concept of **multi-modality**, or modality shifting, consists in support of a mechanism of switching between different Planning Modes depending on a current context. The multi-modality concept can be related to the ideas of Qian et. al [31] and Mehta et. al [27] that utilise Partially Observable Markov Decision Processes (POMDP) model to generate navigation control policies. However, unlike CoHAN, no situation assessment is implemented in these works. The situation assessment and mode shifting scheme between three primary Planning Modes was formerly introduced in the earlier version of HATEB Local Planner [36]. The HATEB version, embedded in CoHAN, is augmented with a fourth mode – **Backoff-recovery**. More details on that can be found in [Chapter 4](#).

When it comes to **Human Path Prediction services**, there are needed to generate socially compliant trajectories for a robot when it encounters dynamic humans in a vicinity. It is possible to derive path prediction directly or to predict a goal and then use it for path prediction. The current human velocity can be used to build a linear prediction, as it is done in [22], or the SFM method can be employed, as it is suggested in [20] and [15]. Authors of [10] explain intention inference by reasoning counterfactually, and [16] presents a geometric-based long-term Bayesian

Human Motion Intentionality Prediction (BHMIP) method, encapsulated in CoHAN system. Another interesting Bayesian framework is a work of Fisac et. al [17] that proposes a probabilistic human model where "rational" human actions viewed as indicators of model's ability to describe the human's current motion.

Finally, the CoHAN's extended version [17] is planned for integration into MARRtino software at the stages of the further development. Its conceptual novelty in comparison with the previous versions of the planner is the invisible humans detection that is done through the understanding of locations from which a human who is currently invisible to the robot can suddenly appear, and then adjusting of the robot's path and velocity in anticipation of potential collisions. This work takes an inspiration from the research activities of Krishna et. al on robot motion safety with regards sudden human emergence [24], [6] and [23].

Chapter 3

Co-operative Human-Aware Navigation Planner

As introduced previously, the Co-operative Human-Aware Navigation (CoHAN) planner is a chosen approach to implement a Social Navigation in our system. The overall architecture of the integrated planner is represented in the [Figure 3.1](#) and described in the current chapter.

The red building blocks of architecture in [Figure 3.1](#) are the CoHAN-specific components that were added to the standard ROS Navigation Stack to enrich it with human-aware properties. These components are:

- **HATEB Local Planner** – a core component of the system which is in its essence a human-aware extension of Timed Elastic Band (TEB) Local Planner, introduced in [Section 2.2](#).
 - **Human State** – a part of Human Tracking mechanism that controls inclusion of the Human Safety and Human Visibility costmap layers (when the Human State is "static", the layers are added, when the Human State is "dynamic" – not);
 - **Planning State** – a criterion for the selection of the Human Path Prediction service;

- **Human Safety and Human Visibility** – layers of both Local Costmap and Global Costmap. The Human Safety adds cost for approaching human too close, and the Human Visibility penalizes the surprise appearances from behind and makes the robot enter the human’s field of view from a larger distance. A design choice to apply these layers to static humans only is motivated by limitations on computational resources (the response time for static humans is usually slow);
- **Human Path Prediction** – a module where requesting a certain service to predict/suggest possible paths of the tracked dynamic humans is performed.

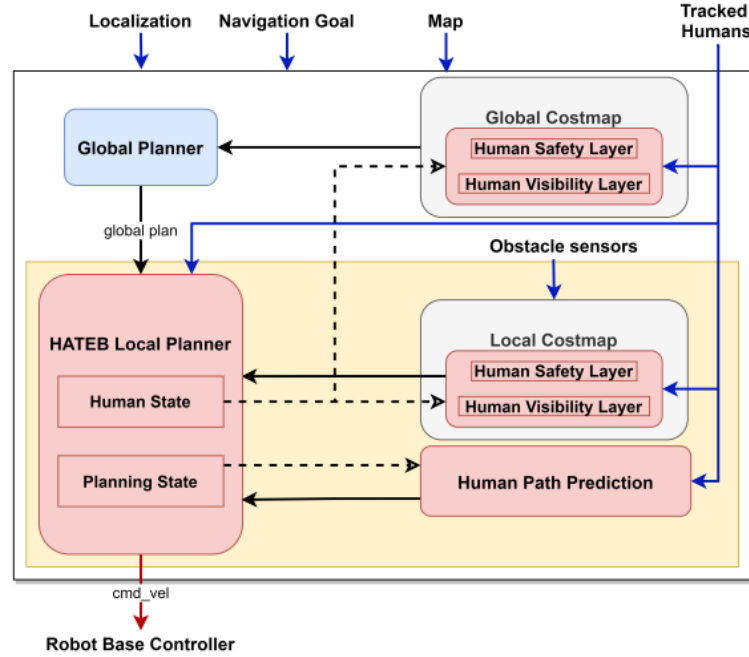


Figure 3.1. CoHAN planner software architecture. Reproduced from [37].

To understand inter-connections between the different components of architecture better, let us consider Human Tracking approach, Human Path Prediction module and the transition scheme between Planning Modes in more detail and then conclude with a discussion of Human-Aware Constraints, encapsulated in the HATEB optimisation scheme.

3.1 Human Tracking

The Human Tracking is provided by an external to the Navigation Stack module (top-right text input in the [Figure 3.1](#)). Although all humans in the environment are tracked, only those within a Visible Region (Field of View, FoV, formed by ray-tracing from the robot's position on the environmental map) are considered visible to the robot. Furthermore, by introducing the Planning Radius (tunable parameter), the Visible Region is narrowed, and only humans within the Planning Radius are considered for planning. To sum up, **while the robot is moving in the environment, the system accounts for only humans within the Planning Radius** that are in the Visible Region. For example, in [Figure 3.2](#) black circles represent humans who are tracked by external component, but not visible to a robot. Then, there are two green circles in yellow zone (Visible Region), but not outside the blue one (region restricted by Planning Radius). They denote humans who are visible, but currently disregarded by the planner.

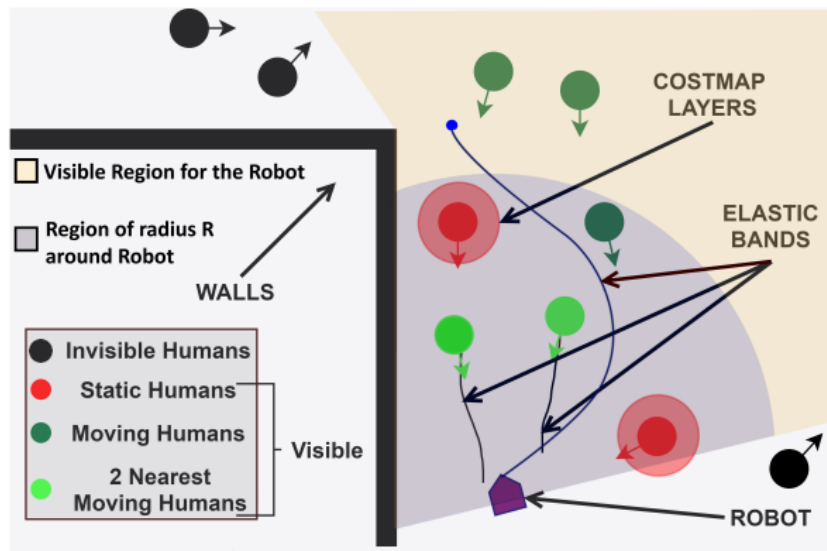


Figure 3.2. Different types of humans, considered in the system. Reproduced from [\[37\]](#).

The humans within the Planning Radius are called observable. Each observable human is classified as "static" or "dynamic", and the classification result is recorded in the Human State. Following that, the `human_layers` costmaps' plugin examines

the Human State and adds the Human Safety and Human Visibility costmap layers around the static humans (red circles in Figure 3.2). The Human Safety layer (Figure 3.3a) is modelled as a 2D Gaussian around the human, and the Human Visibility layer (Figure 3.3b) – as a 2D half Gaussian on the backside of the human. Both these layers have a cutoff radius (tunable parameter) beyond which the cost is zero.

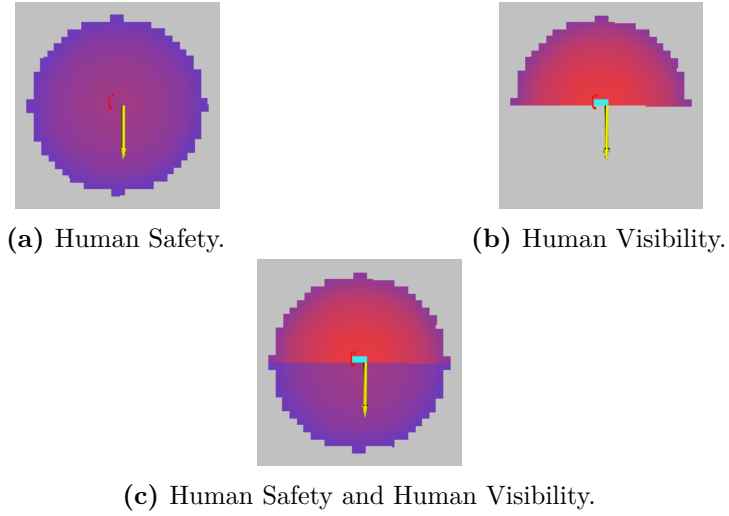


Figure 3.3. Costmap layers around the static humans, displayed in RViz. The yellow arrow denotes a human pose. The Human Safety layer (blue) is a 2D Gaussian around the human, and the Human Visibility layer (red) is a 2D half Gaussian on the backside of the human. Both these layers have a cutoff radius beyond which the cost is zero.

When it comes to dynamic humans (among those who are observable), the system detects two nearest of them (light green circles in Figure 3.2), attaches elastic bands, predicts their paths and plans their possible trajectories until they move behind the robot or out of the Planning Radius. The planning is restricted to the two nearest humans by design choice because extending it beyond two humans would not yield a real-time control of the robot since the Local Planner executes a computationally expensive optimisation in each control loop. The prediction of human paths is performed by the corresponding module using one of the described in the following section services.

3.2 Human Path Prediction

The Human Path Prediction module is responsible for the **generation of the global plans for the two nearest dynamic humans** in observable region. Once the global plans are generated, they are handed to the HATEB Local Planner to build the local plans.

There are four different prediction services, and which one of them is currently in charge depends on the system configuration and active Planning Mode. The services are:

- **PredictBehind** — predicts that the human goal is behind the robot using the position of the robot at the moment when the human enters zone within the Planning Radius. Based on the goal prediction, the path prediction is then obtained. The service is called when the robot is in **DualBand** mode;
- **PredictGoal** — predicts the most probable human goal among the set of goals, provided to the system manually. Based on the goal prediction, the path prediction is then obtained. The service is called when the robot is in **DualBand** mode as alternative to **PredictBehind**;
- **PredictVelObs** — does not predict any human goal. The path prediction is built by extrapolating the current human velocity over a fixed duration window (tunable parameter). The service is called automatically when the robot switches to **VelObs** mode (default service of this mode);
- **PredictExternal** — does not predict any human goal, but accepts it from an external entity. Based on the provided goal, the path prediction is then obtained. The service can be called when the robot is in a **DualBand** mode and can be set active together with **PredictBehind** or **PredictGoal**.

3.3 Planning Modes

As it was mentioned in [Chapter 3](#), one of the most important advantages of the CoHAN planner is the ability to handle multi-context. This property of adaptability is provided by introducing different Planning Modes and intelligent mechanism of switching between them because in the intricate human-robot contexts the planning problem cannot be solved using just a single mode. In a nutshell, the multi-modality mechanism implemented in CoHAN is the following:

1. The system takes as input the navigation goal;
2. The **continuous process** starts:
 - (a) HATEB Local Planner accesses human-robot scenario and determines the Human State and the Planning State;
 - (b) Depending on the value of the determined states, planner shifts between different Planning Modes;
 - (c) The current Planning Mode is used to choose the command velocity to send to the robot's base controller;
3. Process completes when either the goal is reached or robot is stuck or in a collision in which case the recovery behaviour is activated.

The available Planning Modes are:

- **SingleBand** — the mode in which the elastic band is added only to the robot to avoid obstacles in the environment. This mode can be seen as a purely reactive mode with social constraints. The planning system starts in **SingleBand** and stays in it as long as there are no humans within Planning Radius. This mode is the least computationally expensive and is meant for use when humans are far from the robot or there are no humans;
- **DualBand** — the mode in which the elastic bands are added to two nearest observable dynamic humans and to the robot. Trajectories for two humans

and robot are optimised simultaneously. The robot’s trajectory is adapted to the motion of humans and their predicted goals. This allows the robot to demonstrate a proactive behaviour. Additionally, the planned for the humans trajectories offer a possible solution for the human-robot conflicts;

- **VelObs** – the mode in which all the human-aware criteria are used during planning, but elastic bands are added to humans only if they have some velocity. This mode is less proactive, but it allows an active re-planning and is useful in crowded scenarios or when the robot cannot move due to the Entanglement Problem of the DualBand mode (Figure 3.4);

***The Entanglement Problem [36]:** HATEB assumes that humans keep moving and tries to adapt its path according to their motion. This assumption can result in an entanglement of trajectories when the human no longer moves, and the robot keeps waiting for the human to move, neglecting the other possible solutions.*

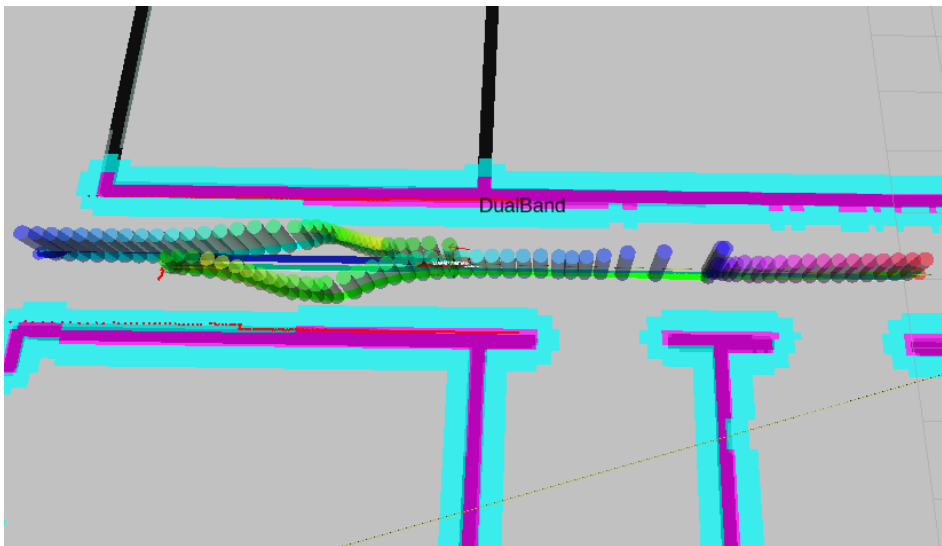


Figure 3.4. Illustration of the situation when the Entanglement Problem would have happened, but switch to **VelObs** happened instead. The blue trajectories belong to the robot, and the green ones – to the human.

- **Backoff-recovery** – the mode that is activated when the robot encounters the Full Blockage Problem of the VelObs mode (Figure 3.5). In **Backoff-recovery** robot moves backwards slowly until it finds a free space to clear the way (by querying the costmaps and setting a temporary goal in the possible direction). Once the robot clears the way, it waits for the corresponding human to complete its navigation or a timeout (tunable parameter) and then proceeds to its goal. It can also accept new goals in the waiting state.

***The Full Blockage Problem:** the robot is in the near vicinity (tunable parameter) of the human, and it is stuck without progressing towards the goal for a time window of a certain duration (tunable parameter). There is no solution to the planning problem unless one of the agents completely clears the way for the other. This kind of situation commonly occurs in a very narrow corridor where only one person (or robot) can navigate at a time. If human and robot face each other in a very narrow corridor or another situation where one of them has to clear the way for the other, the system gives priority to the human and makes the robot clear the way.*

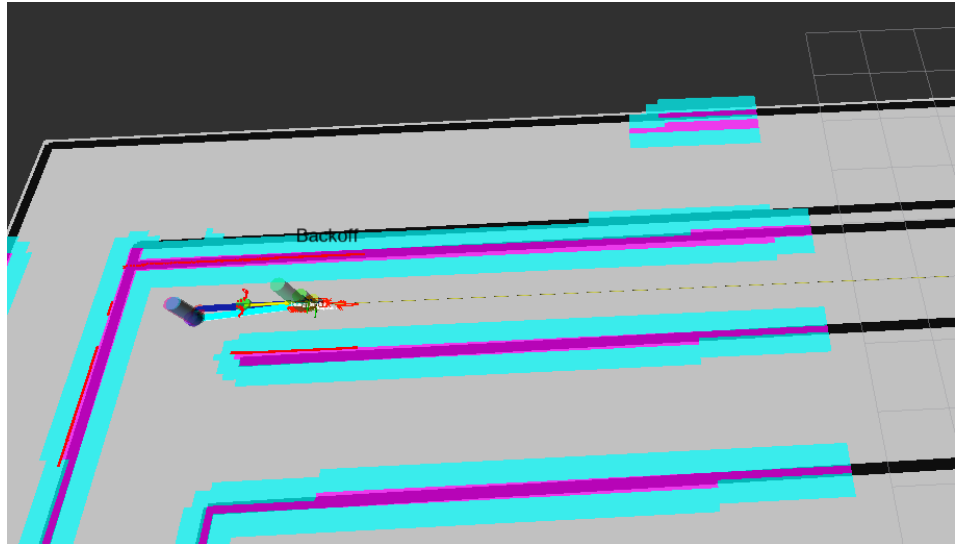


Figure 3.5. Illustration of the situation when the Full Blockage Problem happened and the robot switched to **Backoff-recovery**. The blue trajectories belong to the robot, and the human has stopped.

Moving on to the decision-making process involved in transitioning between different modes, the full scheme of possible shifts is shown in [Figure 3.6](#). The shifting criteria are thoroughly explained in [\[36\]](#) and [\[37\]](#). On a low level, they are based on a thresholding of the measured human-robot distances and human velocities depending on the states of observable humans. Conceptually, switching `SingleBand` \leftrightarrow `DualBand` is introduced to handle dynamic humans in the environment, `DualBand` \rightarrow `VelObs` is intended to solve the Entanglement Problem and `VelObs` \rightarrow `Backoff-recovery` addresses the Full Blockage Problem. Then, `SingleBand` \leftrightarrow `VelObs` is possible in two cases: when entanglement occurs immediately and when `DualBand` is manually disabled. Disabling is recommended for in the crowded environment where `DualBand` shows to be less effective than `VelObs`. Finally, the only exit from a `Backoff-recovery` state is through the `Backoff-recovery` \rightarrow `SingleBand` transition that happens under condition of absence of dynamic humans within Planning Radius. This is the limitation of the current implementation.

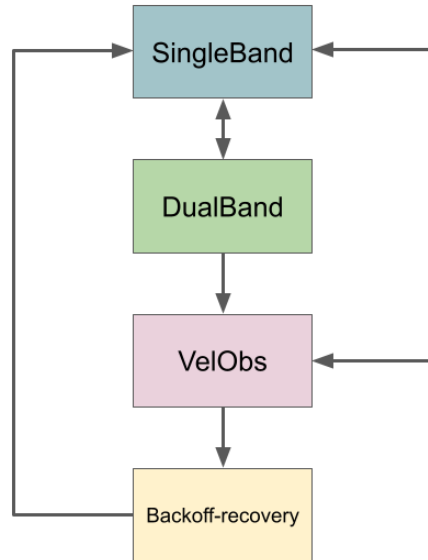


Figure 3.6. Mode transition scheme. Arrow represents one-sided transition, double arrow – two-sided transition.

3.4 Human-Aware Constraints

HATEB Local Planner takes all activated human-robot and kinodynamic constraints and plans the trajectories of the robot and humans. Besides Human Safety and Human Visibility costmap layers, HATEB uses several Human-Aware Constraints in its optimisation scheme for proactive and legible planning around humans in the environment.

Several of these constraints were implemented in the previous versions of the planner and are now deactivated. For example, **Modified Robot Velocity constraint** – a nonlinear profile for the velocity of the robot which slows down the robot up to 75% in the close vicinity of the human; **Time-To-Collision (TTC) constraint** – the time it would take to the robot to collide with the human from the current position and with current velocity; **Modified Time-To-Collision (TTCplus) constraint** – TTC with regulated addition of the error to the optimisation so as to decrease the number of false negatives. Although disabled, these constraints are present in the framework to be used as control mechanisms to leverage optimisation when needed. The reason for deactivation was the addition of a new constraint with similar effects – the Relative Velocity constraint. The **Relative Velocity constraint** adds cost to the optimisation based on the relative velocity between humans and the robot and their positions. The cost is computed as:

$$cost_{rel_vel} = \frac{\max(\vec{V}_{rel} \cdot \vec{P_r P_h}, 0) + \|\vec{V_r}\| + 1}{\|\vec{P_r P_h}\|} \quad (3.1)$$

where $\vec{P_r}, \vec{V_r}$ – position and velocity of the robot, $\vec{P_h}, \vec{V_h}$ – position and velocity of the human, $\vec{V_{rel}} = \vec{V_r} - \vec{V_h}$. The effects of the constraint are: 1) If the robot can find a path with a greater distance from a human, it chooses that path with a normal velocity profile; 2) If robot cannot find a path with a greater distance from a human, it is commanded with a low velocity in the close vicinity of the human; 3) Early intention show (robot starts moving in the intended direction early).

Chapter 4

Implementation

In this chapter, the MARRtino robot software is described first, and then the integration process is demonstrated.

4.1 MARRtino software

The MARRtino robot is a low-cost nevertheless fully compatible with other expensive and professional platforms ROS-based differential-drive mobile robot that is an open hardware/software project, designed for experimentation and commonly used in education and research.

The software for MARRtino exists in several forms, starting from low-level source codes and ending with a ready-to-use pre-installed on a VirtualBox Virtual Machine version. The [marrtino_apps](#) repository is a **Docker-based** MARRtino software that was used for the purposes of this thesis. It contains ROS(kinetic/melodic)-packages, representing different robot functionalities and control/visualisation modules, distributed among **Docker images** (or, as runtime instances, **Docker containers**) and interfaced with Python and other languages.

The available images are:

- System support:
 - `orazio` (for the real robot)
 - `stage_environments` (for the `stage` container)
 - `nginx`
 - `devrt/xserver`
- Operational:
 - `base` (robot base)
 - `system`
 - `teleop` (teleoperation)
 - `navigation` and `navigation-cohan`
 - `mapping`
 - `vision`
 - `speech`
 - `objrec` (object recognition)

In particular, the `navigation/navigation-cohan` and `mapping` components are responsible for the Navigation Stack functioning. Using nomenclature of [Section 2.1.2](#), `mapping` contains the Mapping module, and `navigation` stores the executables to perform Localisation and to move robot base along the paths, provided by the Path Planning. The Path Planning itself is represented by the variety of planning approaches. They are normally implemented as external systems and pulled from the outer repositories into inherited from `navigation` image. Similarly, the Co-operative Human-Aware Navigation (CoHAN) planner repository [CoHAN_Planner](#) is pulled to the `navigation-cohan` image. Thus, this image becomes a foundation for the overall integration process.

The Docker containers, involved in the integration and testing of CoHAN, are the following five:

- `navigation`
- `base`
- `stage`
- `nginx`
- `xserver`

The last two containers in the list are related to the computer setup (web server), `base` and `navigation` are the runtime instances of the `base` and `navigation-cohan` images, respectively, and `stage` is a simulator. Because no changes were made to the Mapping process, the `mapping` container is not included in this list, even though it is one of the building bricks of a ROS Navigation Stack.

4.2 Planner Integration

When Docker has started and all of the required containers are running, the next step of the pipeline is to start executing the commands inside these running containers. To launch the navigation, the containers of interest for the execution of the commands are the `stage` and the `navigation`. Hence, in order to integrate and challenge a new planner, these two containers are the architectural components that must be modified.

4.2.1 Stage Container

MARRtino ROS nodes are fully compatible with the Stage [3] – the standard ROS 2D simulator, and the corresponding MARRtino software package contains a Python script to automatically create and run the simulated environments with human/robot agents.

Mainly two modifications were made in the **stage** container:

1. The map collection was complemented;
2. The Human Tracking system was provided.

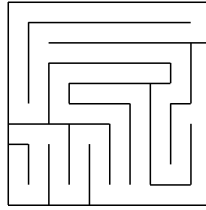
First, some of the existing maps were customized and/or augmented with semantic information, while others were created from scratch, because a large collection of maps is required to conduct an extensive testing. On the one hand, the task was to model all of the intricate scenarios that the planner was originally designed to handle efficiently (e.g. doorways, hallways, corridors) in order to verify its declared properties. On the other hand, since enhancement of a robot's reliability in a healthcare environments is our focus, the aim was to learn how to account for the specificity of a domain the best by simulating medical contexts. [Figure 4.1](#) shows four main maps, used in experiments: labyrinth, Emergency Department, Intensive Care Unit (ICU) and Intensive Care Unit with the wall in the middle (ICU2).

Second, the Co-operative Human-Aware Navigation planner's packages are built as plugins to the standard 2D Navigation Stack and hence they follow the same topics to publish navigation goals. However, as it is emphasized in [Section 3.1](#), there is an external system in charge of Human Tracking. According to its arrangement, to include humans into the system, they must be published on a `/tracked_humans` topic following the particular message structure. This is implemented in a Python script, called `humans_bridge.py` that requires as input the number of humans and pipes a subscriber to the humans and a publisher to `/tracked_humans`. The script must have been added to the **stage** to function properly.

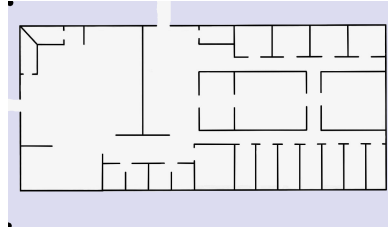
4.2.2 Navigation Container

Finally, after Human Tracking is activated and the Localisation (e.g. the basic `amcl`) is launched, the core component of the Navigation Stack with the updated version of `move_base` node must be launched. The CoHAN main executable file

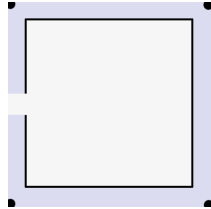
is called in our architecture the `cohan_nav.launch`. It was complemented with necessary frame transformations to eliminate naming inconsistencies and placed in the `navigation` folder together with new configuration files containing the `move_base`'s, Local Costmap's, Global Costmap's and HATEB Local Planner's parameters. The `cohan_nav.launch` uses HATEB as a Local Planner in `move_base` node and, besides this node, contains Human Path Prediction and Human Filtering.



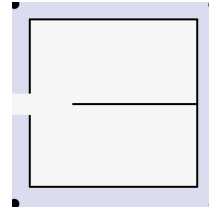
(a) Labyrinth.



(b) Emergency Department.



(c) ICU.



(d) ICU2.

Figure 4.1. Map files used to simulate the stage.

Chapter 5

Experiments

This chapter is devoted to the experiments that have been conducted to create a complete characterisation of the integrated system and assert enhancement of the robot reliability by means of a chosen Human-Aware Navigation Planning approach. The first part briefly describes the experimental setup. Then, the results in various simulated human-robot contexts are presented and thoroughly analyzed qualitatively and quantitatively followed by a short statistical analysis of the system evaluation by clinicians in terms of acceptability and usability in a real environment.

5.1 Experimental Setup

The system has been tested in a simulated environment. The real-life tests would be required to validate a mature system. However, because the proposed integrated framework is in its early stages of development, it must first be debugged in a simulation. The ROS Stage simulator and RViz visualisation tool were used to model the navigation scenarios, and the `rqt_reconfigure` plugin – to tune the navigation parameters at runtime.

The final system configuration can be found in the [Appendix](#). Some of the parameters' values are kept as original, the other ones (e.g. inflation radii, minimum distance to static obstacles, minimum human-robot distance) widely vary throughout

the testing process in order to check the hypotheses about the presence or absence of the certain features or in attempts to find an explanation of a particular robot behaviour.

Then, another essential aspect in the testing pipeline is the simulation of the human motion. The default ROS Stage in MARRtino does not provide utilities for the intelligent dynamic human simulation, although human-awareness of the robot must be evaluated in experiments including moving human agents.

In this work, we use mainly two approaches to human dynamics modelling: publishing on a human velocity topic or publishing on a human pose topic. In simple cases, when a linear motion (i.e. the trajectory is a straight line) is sufficient because the space is limited (e.g. in the narrow corridor) or when a circular motion with a constant angular velocity around z axis satisfies the simulation goal, a basic publisher on a human command velocity topic is employed. Otherwise, when nonlinear human motion simulation can be helpful to better challenge the planner (e.g. in the wide corridor), special Python scripts are used. One of them contains the routine that precomputes the human trajectory and then continuously publishes it on a human pose topic. Another one is a complementary script, developed to model the situation of emergency in the Intensive Care Unit. In this script an incremental publishing on human pose topic is exploited to, first, align the human orientation with the direction of the future motion and, then, move linearly along the line connecting human current and goal positions. This is done simultaneously for the eight predefined humans, and the goal position is the same for all of them so to create a crowd in a goal vicinity.

Now, with understanding of the simulation tools, we present the results in diverse simulated human-robot contexts.

5.2 Results

In this section, we examine in detail the robot's ability to socially navigate within the family of specially designed scenarios with different characteristics and of varying levels of complexity:

- **Visibility Test** – [section 5.2.1](#)
 - A human in open space
 - A human and a wall
 - Two humans in open space
- **Door Crossing Scenario**
 - Wide Doorway
 - Narrow Doorway
 - Bed Approach
- **Narrow Corridor Scenario**
 - A "never stopping human"
 - A human who stops
- **Wide Corridor Scenario**
 - Free Corridor
 - Cluttered Corridor
- **Crowded Scenario**
 - Free Space Crowd
 - Emergency Situation

5.2.1 Qualitative Results

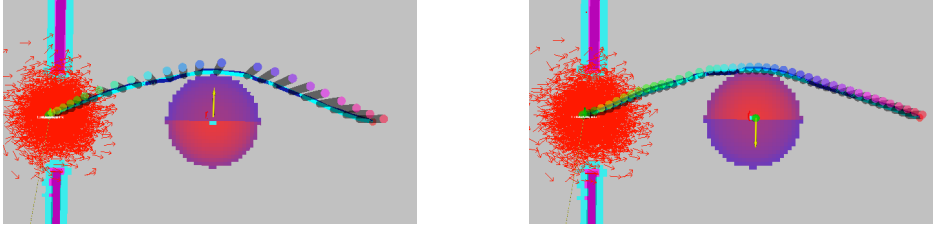
In all the snapshots from RViz: dark and light blue lines are the global path and elastic band (local path) of the robot, dark and light green lines are global paths and elastic bands generated for humans, yellow arrows indicate human orientations, and the trajectories of the robot and humans are shown as coloured bars/dots. They are the poses planned by HATEB Local Planner, and the colour visualises the timestamp. If the colour of the predicted human pose bar is the same as the colour of the robot pose bar – they will both be at that location at the same time.

Visibility Test

This test considers the presence of only static humans and is intended to estimate the influence of a particular component of the system – Human Visibility layer (Human Safety is implicitly verified whenever static humans are present and the robot does not collide with them). In particular, the hypothesis of interest is **if the visibility cost indeed helps to prevent sudden emergencies of the robot from behind the human**.

We start from a simple test of a single static human in the open space. Since the human is static, the robot is always in `SingleBand`, and the system adds the `human_layers` to the costmaps. As it can be seen from [Figure 5.1](#), sometimes robot can choose to pass in front ([Figure 5.1a](#)) and sometimes – behind ([Figure 5.1b](#)) the human. This is explained by the fact that the cost of passing behind or in front is the same outside the safety radius (1.5 m), and since the test is performed in the open space, there are the cells of the same cost from both sides. In other words, this is a costmap-based implementation and the situation is symmetric. The robot has enough space behind the human so as not to disturb him/her and still continue to its goal. **The planner does not make the robot choose the frontal side always – only when the back side is constrained and the robot has to move close to human**. However, even in case when the robot appears from behind

the human, it enters the human's field of view from a larger distance.



(a) The robot chooses to pass in front of the human. (b) The robot chooses to pass behind the human.

Figure 5.1. Visibility Test: A static human in open space. RViz snapshots.

The explanation is further confirmed in tests with one static human standing next to the wall and two static humans standing next to each other in the open space environment.

In case of a static human and a wall, it can be noticed from comparison of [Figure 5.2a](#) and [Figure 5.2b](#), that regardless of human orientation, if there is a free space between the human and the wall, the robot navigates through this space, otherwise – around the human, as in [Figure 5.2c](#) and [Figure 5.2d](#). When there are two humans next to each other in open space, no matter if they are co-directed or contr-directed (in which case they can be considered interacting), if there is a free space between them, the robot moves through this space ([Figure 5.3a](#), [Figure 5.3b](#)), otherwise, it moves around ([Figure 5.3c](#), [Figure 5.3d](#)) because the passing in front is not forced by the costmaps. Interesting question is whether the situation, illustrated in [Figure 5.3b](#), should be counted as problematic. From one side, usually in Social Navigation when two humans are facing each other, the robot should not pass between them since humans are assumed in this case interacting. From the other side, the distance between these humans is more than 3m.

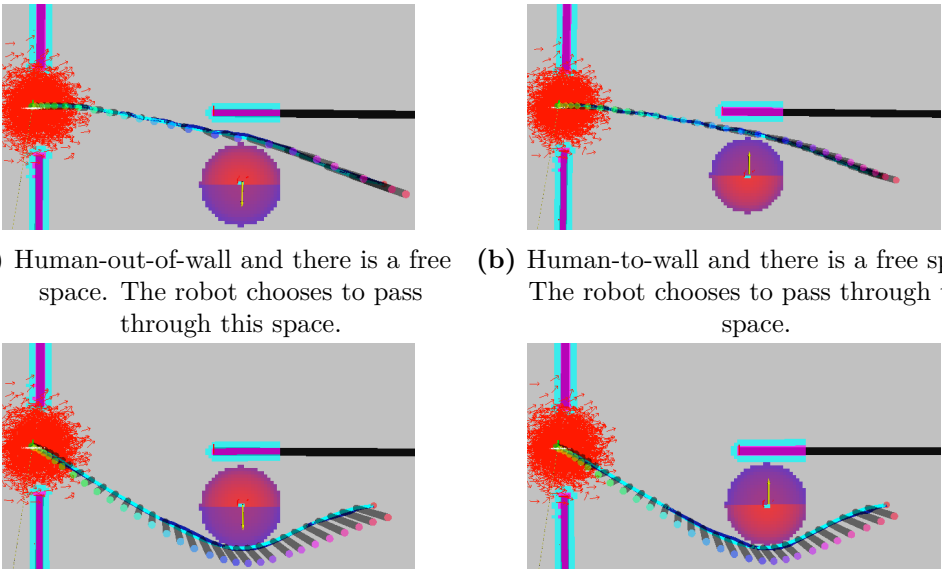
- 
- (a) Human-out-of-wall and there is a free space. The robot chooses to pass through this space.
- (b) Human-to-wall and there is a free space. The robot chooses to pass through this space.
- (c) Human-out-of-wall and there is no free space. The robot chooses to pass around the human.
- (d) Human-to-wall and there is no free space. The robot chooses to pass around the human.

Figure 5.2. Visibility Test: A static human and a wall. RViz snapshots.

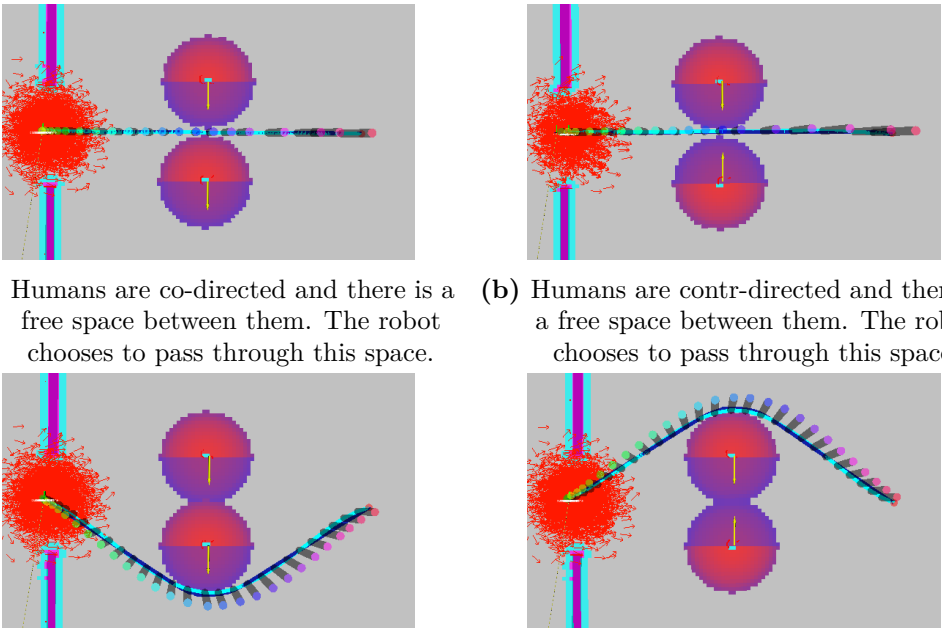
- 
- (a) Humans are co-directed and there is a free space between them. The robot chooses to pass through this space.
- (b) Humans are contr-directed and there is a free space between them. The robot chooses to pass through this space.
- (c) Humans are co-directed and there is no free space between them. The robot chooses to pass behind one of the humans.
- (d) Humans are contr-directed and there is no free space between them. The robot chooses to pass behind one of the humans.

Figure 5.3. Visibility Test: Two static humans in open space. RViz snapshots.

Door Crossing Scenario

The Door Crossing scenario is a series of three simulated situations: Door Crossing with a Wide Doorway, Door Crossing with a Narrow Doorway and a Bed Approach Test.

The first two are the common situations in many human environments including medical institutions, because they are represented by a classic human-robot door crossing setting and can occur at the entrance of any room. In both cases, the simulated environment is a part of the map of Emergency Department (ED) that consists of two rooms connected with the opened door. There is a dynamic human in one room and a robot – in another one. They are placed symmetrically at the same distance to the door. Dynamic human is moving with a constant linear velocity along the straight line, and the goal of the human is somewhere on this line behind the initial position of the robot. Therefore, the `PredictBehind` Human Path Prediction service is activated. The goal of the robot is shown in the [Figure 5.4](#).

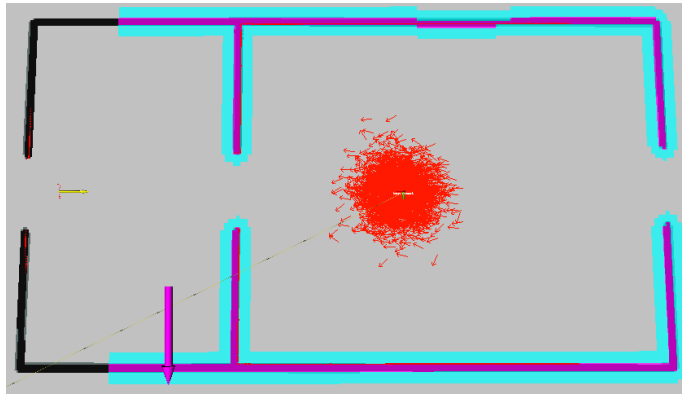


Figure 5.4. Door Crossing: Wide Doorway. Initial setup where the robot goal is denoted by the pink arrow. RViz snapshot.

In a Door Crossing: Wide Doorway case a doorway is wide enough for two agents. This scenario is appropriate to test the robot in a `DualBand` mode because there is a dynamic human, and neither the Entanglement, nor the Blockage Problem can happen unless the human abandons his/her goal and/or stops in the doorway. The tests consistently complete with the success: the robot avoids collisions and

demonstrates the declared behaviour. The robot always correctly switches from a `SingleBand` to a `DualBand`, but the other events vary depending on the run.

During the perfect run, illustrated in [Figure 5.5](#), the robot is able to "guess" the linear motion of the human and plan its own path with a deviation maneuver to let the human pass first. In this case the robot not only avoids the collision, but also makes the greater effort than a human.

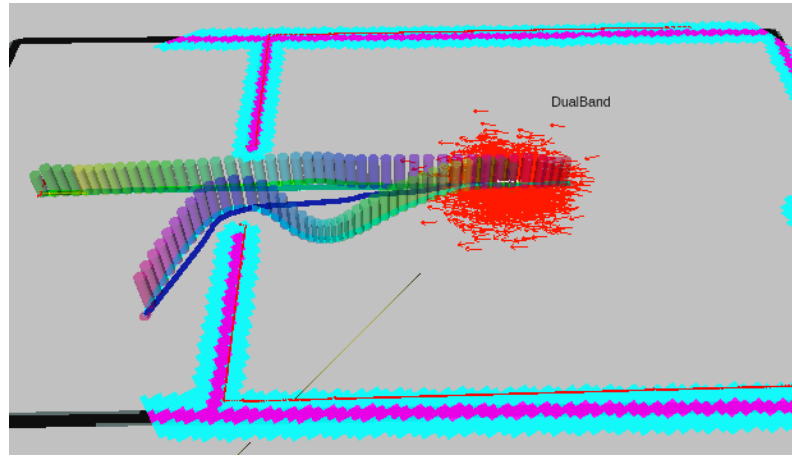


Figure 5.5. Door Crossing: Wide Doorway. Perfect run. The robot's elastic band suggests more effort for a robot. RViz snapshot.

Sometimes even during the perfect run, the situation, demonstrated in the [Figure 5.6](#), occurs: the robot's Local Planner proposes elastic band that crosses the wall. Currently this is one of the minor bugs in the system. However, it does not cause any fatalities as such local plan is never executed: the robot executes only the first pose of the plan and then it re-plans. Furthermore, the system can be tuned to reduce the chance of this bug to appear by increasing the inflation radius.

Finally, in case of a less lucky run, the robot does not plan the greater effort for itself because it assumes human cooperative (i.e. assumes that the human will move aside too). Fortunately, this does not cause collisions since in this case the robot finds an alternative to waiting solution: to pass through the cell of a doorway that is not occupied by the human, as it is presented in the [Figure 5.7](#).

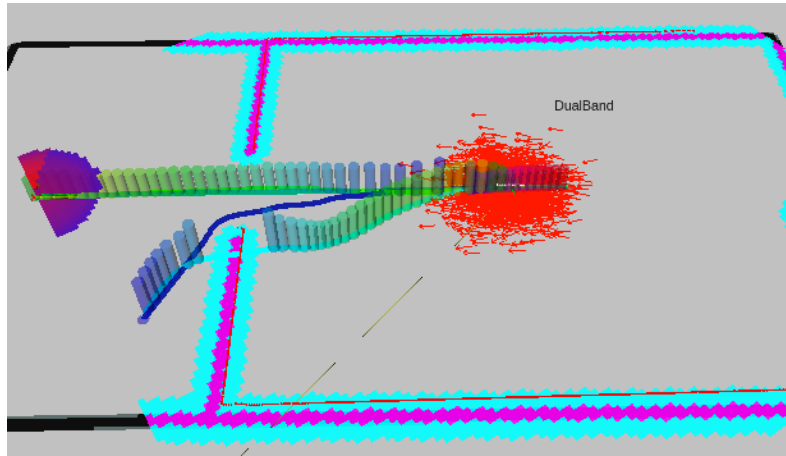
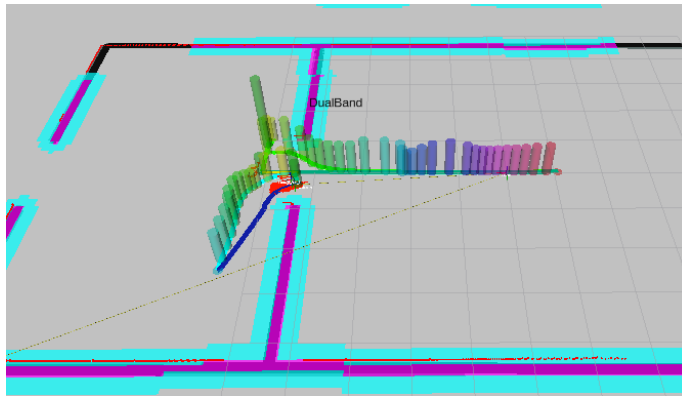
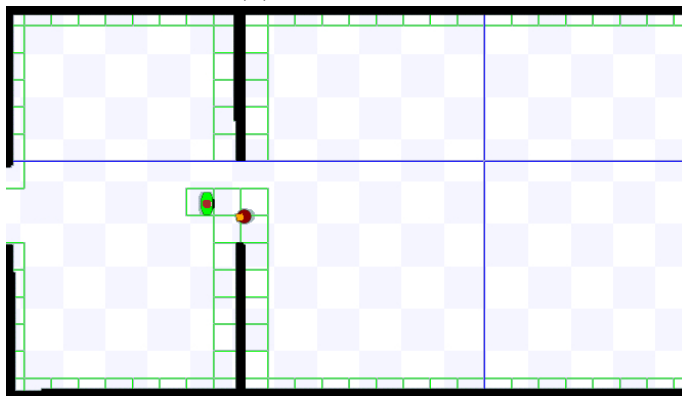


Figure 5.6. Door Crossing: Wide Doorway. The robot's band crosses the wall. RViz snapshot.



(a) RViz snapshot.



(b) Stage snapshot.

Figure 5.7. Door Crossing: Wide Doorway. Simultaneous crossing by the robot and the human. Human is assumed cooperative.

The similar problem is present in a Door Crossing: Narrow Doorway scenario where the setup is the same, but the doorway is wide enough for one agent only. The tests of this scenario are also successful, and the robot behaves in them as declared. However, this task is more complicated because the simultaneous crossing is not possible and the process of entering the doorway itself requires the higher precision. As the result, there is no perfect case for this scenario: the robot does not ever "guess" the human's linear motion and does not plan maneuver to move aside. Instead, it again assumes human cooperative and hence proposes the cooperative solution. Then, at the moment right before the collision (shown in Figure 5.8), the robot takes an action for avoidance – starts moving backwards to let human pass. The robot keeps backtracking until the doorway is free. Eventually, it re-plans and proceeds to the goal in a `SingleBand`.

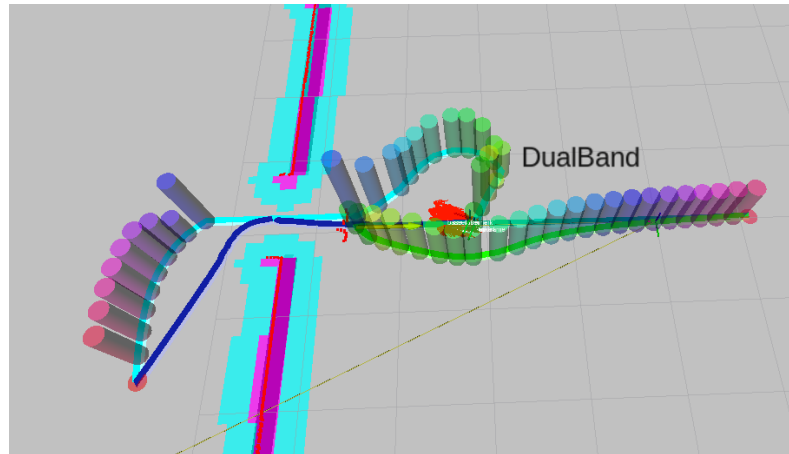


Figure 5.8. Door Crossing: Narrow Doorway. Human-robot crossing. Human is assumed cooperative, and the robot plans to move backward. RViz snapshot.

The robot's planner recommends the cooperative solution in both Door Crossing scenarios partly due to inaccuracy in the human model (maximum velocity and acceleration), but primarily due to this being a property of the Co-operative Human-Aware Navigation (CoHAN) planner by design: it assumes that both agents are interested in cooperation. The system does not predict the human's path, but rather plans it based on estimated

goal and the human model. This is a proactive planning approach, implemented in CoHAN. It is not very accurate, but the robot's behaviour is enhanced in comparison with non-social planners. The "cooperation factor" can be changed using different combinations of settings. Nevertheless, the elimination of the inconsistencies points the directions for the future works, such as improving the goal prediction and updating the human model.

The series of experiments, performed to test the robot's capabilities in human-robot collaborative crossing context, ends with a scenario that is especially relevant in a health-care sector – the Bed Approach scenario. The map of an Intensive Care Unit with a wall in the middle of the room and beds, placed along the walls, is used. The Door Crossing situation is modelled by two agents conflicting for exiting/entering the free space between two hospital beds: the human provider is exiting and the robot is entering (Figure 5.9). The goals of both agents are behind each other (again, the PredictBehind Human Path Prediction service is set on). The "doorway" is wide enough for two agents, but, taking into account the human's location and the inflation radius of beds, the safe simultaneous crossing by the robot and the human is not possible. The expected behaviour of the robot is to wait and let human pass first.

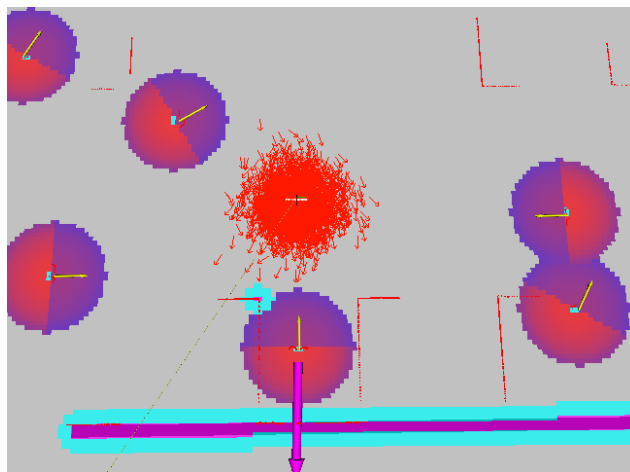
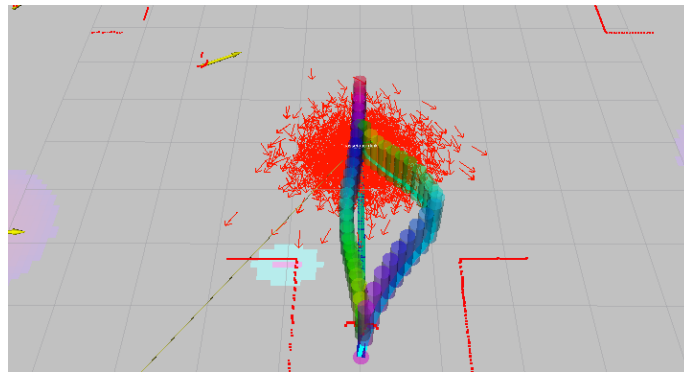


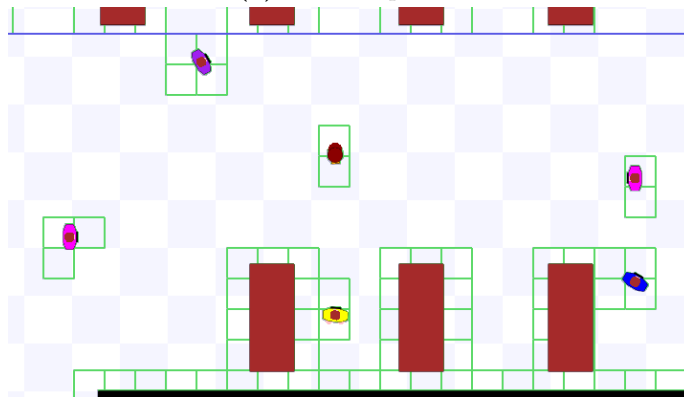
Figure 5.9. Door Crossing: Bed Approach. Initial setup where the robot goal is denoted by the pink arrow. RViz snapshot.

The test is successful because the robot collision never happens and, normally, the human collision does not happen too because the robot demonstrates the expected behaviour:

1. The human and the robot move towards each other. As soon as the robot detects the human, it activates the **DualBand** mode and derives path predictions for itself and for human while imposing more effort on itself (Figure 5.10);
2. The more human-robot distance reduces, the more the robot slows down. When the distance becomes critically small, the robot moves a bit backwards, stops and remains in this state until the human passes it by;
3. The robot resumes to its goal.



(a) RViz snapshot.



(b) Stage snapshot.

Figure 5.10. Door Crossing: Bed Approach. The robot's elastic band suggests more effort for a robot.

Narrow Corridor Scenario

The Narrow Corridor Scenario can happen in any hospital corridor, and it challenges the planner with the Blockage Problem: a long corridor has to be traversed by two agents in opposite directions, and the corridor is wide enough only for a single agent, one of the agents must go back and wait for the other to cross. When one of the agents is a robot, it should back-off giving priority to the human while taking legible actions. Thus, the expected behaviour of the robot is the following:

1. The robot switches from **SingleBand** to **DualBand** and plans a certain trajectory;
2. The robot's way is blocked by the human and the system shifts to the **Backoff-recovery** mode (the **Backoff-recovery** is supposed to be activated when the robot is in **VelObs** mode in the near vicinity of the human ($<2.5\text{m}$), and it is stuck without progressing towards the goal);
3. The robot clears the way for the human, drives away from the corridor and waits on the side until human crosses the robot;
4. The robot resumes to its goal in **SingleBand** mode.

The robot's goal is depicted in the [Figure 5.11](#).

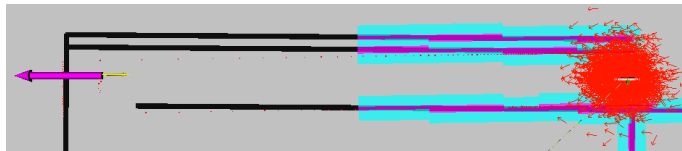


Figure 5.11. Narrow Corridor. Initial setup where the robot goal is denoted by the pink arrow. RViz snapshot.

The dynamic human is moving with a constant linear velocity along the straight line and is a non-collaborative human (does not move back). We consider two different approaches to modelling this human's behaviour: one is the human who never stops,

and, alternatively, the human who moves, but then stops. Both variations are interesting to highlight certain features of the robot behaviour.

A "never stopping human" keeps moving in the corridor even if there is a robot on the way. In real life in the most of the situations human will not be "never stopping". However, this situation can occur, for example, when a person is looking to the phone screen and hence does not see that there is a robot in the corridor.

The test always results in a human collision. Right before the collision, the robot slows down, attempts to move backwards, but at some point it abandons attempts and takes a hit. This happens because **the robot stays in DualBand mode ever since it detects a dynamic human** (Figure 5.12): the condition for a shift to `VelObs` – the human stopping – is never satisfied. In turn, without shift to `VelObs` a shift to `Backoff-recovery` cannot be performed as well. This negative outcome is not critical because the robot does not cause the danger. Yet, the test case demonstrates an important limitation of the CoHAN planner.

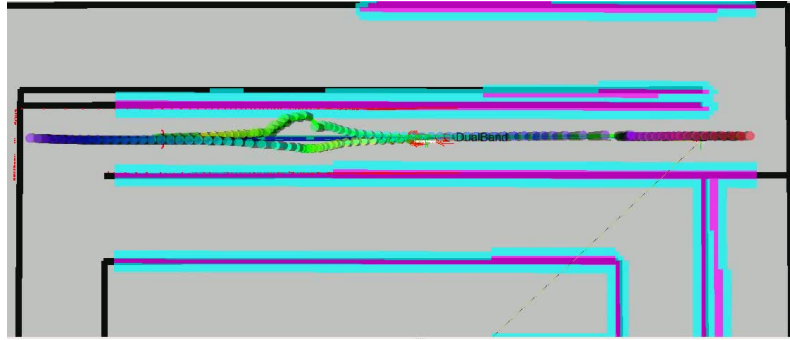


Figure 5.12. Narrow Corridor: A "never stopping human". The robot remains in DualBand ever since it met human. RViz snapshot.

In another variation of the scenario, some time after the robot detects the dynamic human, the human fully stops. In this case, the expected behaviour is not confirmed in experimentation too. The robot gets stuck either without a shift `VelObs` \rightarrow `Backoff-recovery` (a shift `DualBand` \rightarrow `VelObs` is performed correctly) or or with a shift, but without going backwards (Figure 5.13). The problem here is

that the functioning of the robot in **Backoff-recovery mode** is **prone to errors** **because the mode implementation lacks robustness**. **This is regarded as one of the algorithm's current limitations** that leaves room for improvement. For the time being the result of the robot stopping in the narrow corridor after detecting the Blockage is considered satisfactory as opposed to the collision.



Figure 5.13. Narrow Corridor: Human who moves, but then stops. The robot switches to Backoff-recovery. A moment right before the robot gets stuck. RViz snapshot.

Wide Corridor Scenario

The test scenario takes after the Narrow Corridor Scenario, but now the corridor is large enough for two agents. Additionally, on one side of the corridor there are rooms with open doors, so it's possible for the robot to drive there and wait until human passes. The human goal is indicated by the red arrow in the [Figure 5.14](#). Since the human intends to turn, the human agent's motion is nonlinear, and the robot's PredictGoal Human Path Prediction service is activated. A typical goal of the robot in this test is behind the initial position of the human and is shown in the [Figure 5.15](#).

There are two modifications of the scenario: Wide Corridor: Free Corridor and Wide Corridor: Cluttered Corridor. The former serves as a sort of a baseline for

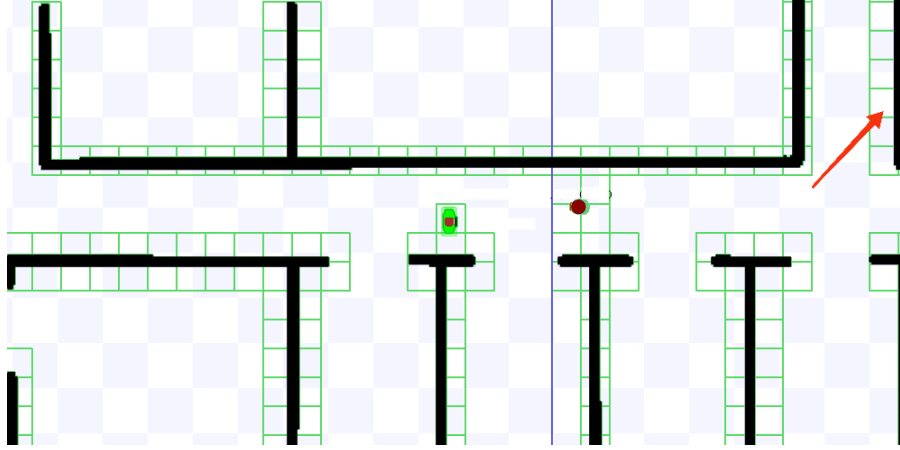


Figure 5.14. Wide Corridor: Free corridor. Initial setup where the human goal is denoted by the red arrow. The Stage snapshot.

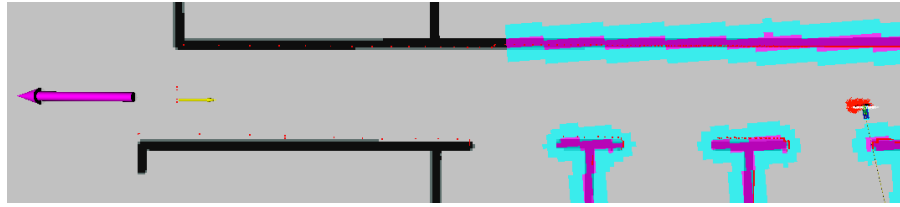


Figure 5.15. Wide Corridor: Free corridor. Initial setup where the robot goal is denoted by the pink arrow. RViz snapshot.

comparisons and is easier to navigate in since it does not contain extra obstacles and hence has more space for manoeuvres.

The Wide Corridor: Free Corridor challenge is usually resolved by the robot positively – similarly to the perfect run in the Door Crossing: Wide Doorway (Section 5.2.1), the collision does not occur and the mode transitions are performed properly. In the perfect run, the cooperative solution that suggests the greater effort for the robot is planned (Figure 5.16). Otherwise, the cooperative solution proposes a heavier load for the human. Sometimes this is justified, as in the Figure 5.17: the moment is close to a collision and the robot’s current velocity does not allow it to perform the maneuver of entering the free room doorway, thus, the robot expects this maneuver to be executed by the human. We can see that the system’s capacity to anticipate navigation conflicts requires an improvement because the situations as in Figure 5.17 can be prevented if the robot foresees them. Additionally, the possibility

to drive off in one of the free room entrances beforehand is completely ignored by the robot. Presumably, the Human Path Prediction is a responsible module of architecture that needs to be upgraded. Furthermore, sometimes the greater effort, planned for the human, should not be justified, but rather treated as a lack of pro-activity on the robot's part, caused by imperfections in the human modelling. For example, the human global plan in the [Figure 5.18](#) is derived correctly, and the overlapping of the robot's and human's global paths can be noticed. However, this observation is not exploited in the decision-making process to make the robot proactively change the side of the corridor. Instead, the human's elastic band suggests the deviation.

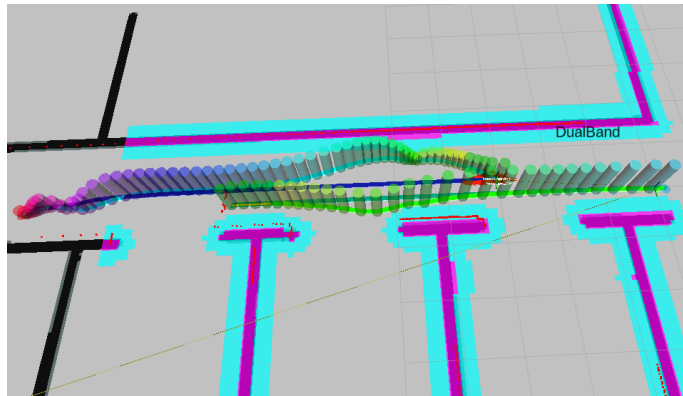


Figure 5.16. Wide Corridor: Free Corridor. Perfect run. The robot's elastic band suggests more effort for a robot. RViz snapshot.

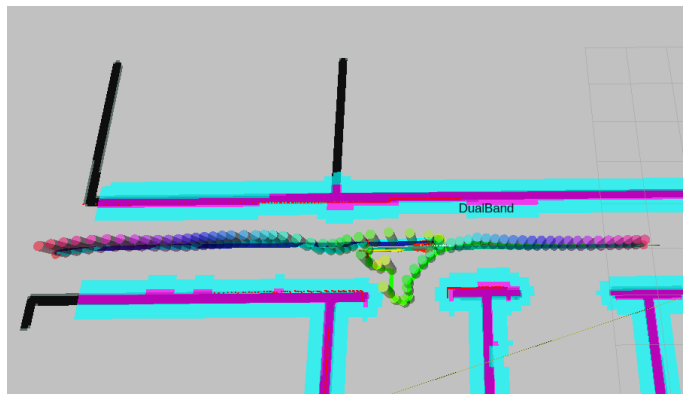


Figure 5.17. Wide Corridor: Free Corridor. The human's elastic band suggests justified extra effort for a human. RViz snapshot.

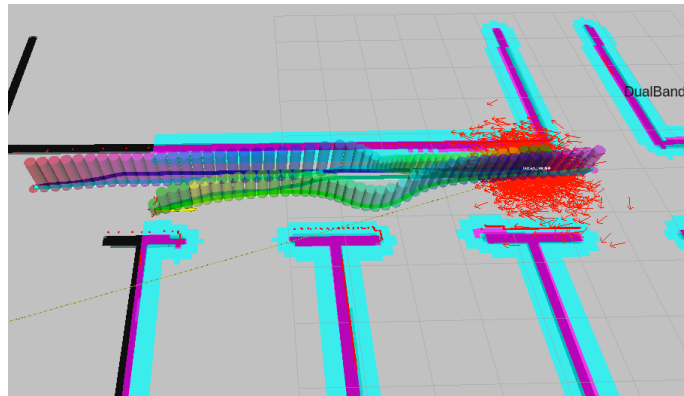


Figure 5.18. Wide Corridor: Free Corridor. The human's elastic band suggests not justified extra effort for a human. RViz snapshot.

Moving on to a Wide Corridor: Cluttered Corridor, the setting is similar, but now a more realistic clinical environment is simulated. The [Figure 5.19](#) shows the hallway of Emergency Department. As it is investigated in [\[40\]](#), patients are often treated in the corridors when the ED bedrooms are full. Placing patients in hallways is a way to handle an overflow of patients. Therefore, the corridors are often overcrowded and cluttered with stretchers and other medical equipment.



Figure 5.19. Emergency services assistant wheels a stretcher down an Emergency Department hallway lined with patients at the hospital. Patients routinely have to wait for an open bed.

We model a simplified version of the Cluttered Corridor scenario where there is only one dynamic human and all the stretchers are leaned against the same wall. Then, even if the corridor is wide enough for two agents, only one agent at the time can pass between the stretcher and the other wall. The robot and the human start navigating from the opposite sides of the cluttered part of the corridor, and their goals are behind each other.

The test normally produces the same series of events at each run:

1. Following its initial path, the robot arrives in the middle of the cluttered hallway where it detects a human and switches from **SingleBand** to **DualBand**;
2. The robot backs-off all way back until it reaches the vicinity of its initial position (which is outside the cluttered zone) and then comes to a halt.

This test result leaves the dual impression. The **robot exhibits a safe behaviour** that enables for collision avoidance. It even **gives priority to a human** and does not insist on pursuing its goal as well as does not abort in the middle of the corridor, but moves backward and clears the way instead. However, unfortunately, both issues, discovered in the Wide Corridor: Free Corridor testing, are confirmed: the **robot is neither proactive nor anticipative enough**. The example of non-proactive behaviour is given in the **Figure 5.20**: there are three options for one agent to clear the way for the other while crossing the cluttered section of the hallway: two free spaces between the stretchers and one free room entrance, nevertheless, the CoHAN system always proposes the deviating elastic band for the human. The statement that the robot is not anticipative, on the other hand, is based on the fact that it only begins to resolve a conflict after it has already encountered it, not in advance.

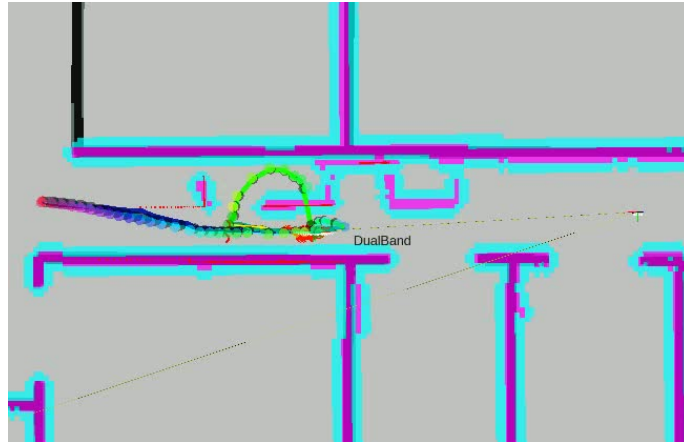


Figure 5.20. Wide Corridor: Cluttered Corridor. The human’s elastic band suggests extra effort for a human. RViz snapshot.

Crowded Scenario

The series of experiments concludes with a Crowded Scenario which is the most challenging for the CoHAN system. The scenario consists of two contextually distinctive sub-scenarios, invented to estimate the robot reliability in a crowd from two different perspectives. Both scenarios are simulated in Intensive Care Unit (ICU) environment.

The first sub-scenario covers the context of a crowd in the free space and is named correspondingly a Free Space Crowd. The map is a room that does not include any static semantic objects other than the beds along the walls. The humans are distributed around the room and form a chaotic crowd. There are 14 humans in total: 10 of them are dynamic and 4 – static. The dynamic humans have different constant velocities and move along the circular trajectory each (Figure 5.21). The robot starts at the room entrance, and its objective is to visit all of the static humans while navigating safely and intelligently through the crowd. Therefore, the robot is tasked with a sequence of four goals (Figure 5.22).

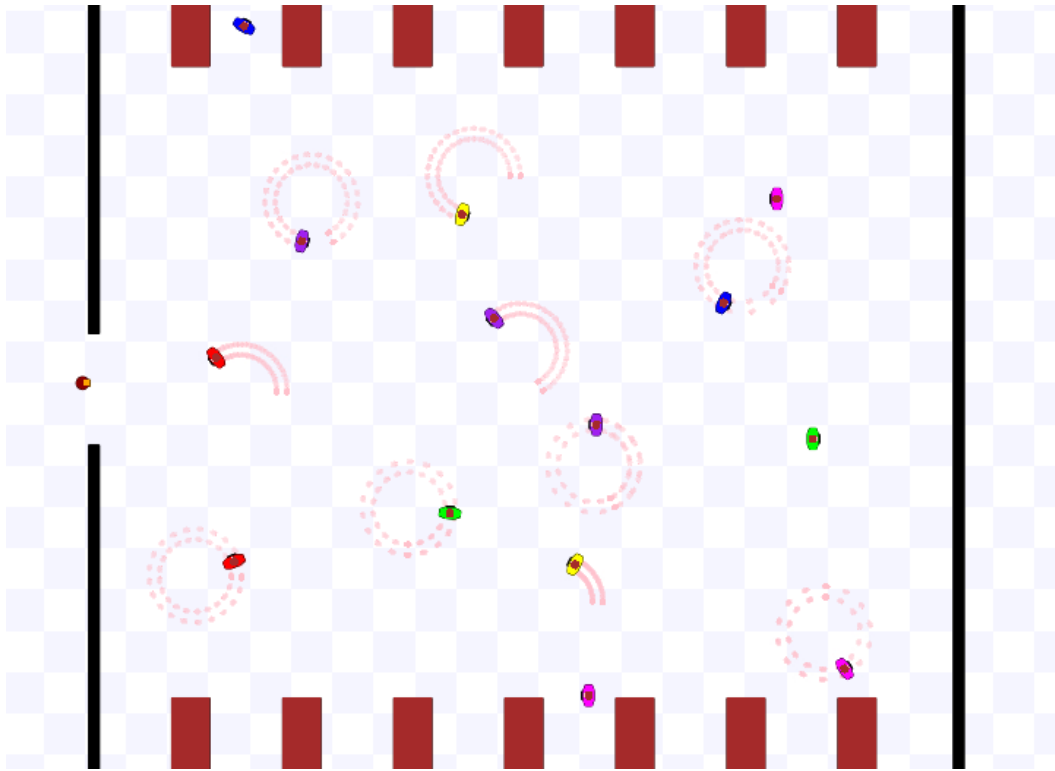
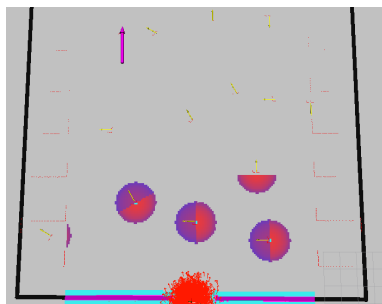
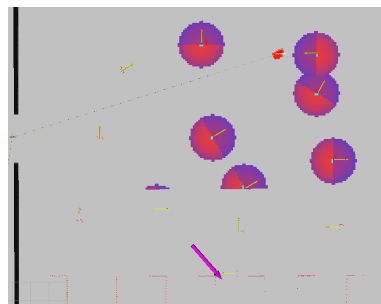


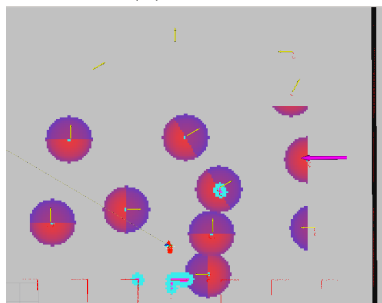
Figure 5.21. Crowded Scenario: Free Space Crowd. Initial setup where the humans' motion is represented by their footprints. Stage snapshot.



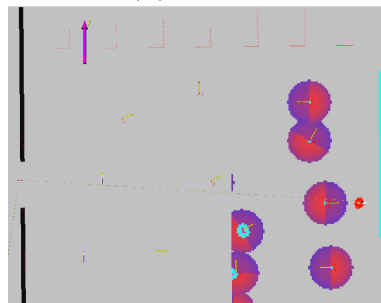
(a) Goal 1.



(b) Goal 2.



(c) Goal 3.



(d) Goal 4.

Figure 5.22. Crowded Scenario: Free Space Crowd. Initial setup where the robot goals are denoted by the pink arrows. RViz snapshots.

The path from the initial position to the Goal 1 is the easiest part of the overall navigation task because the generated robot's global path almost does not interfere areas occupied by dynamic humans. The most evident positive aspect of navigation in the crowd can be observed from the [Figure 5.23](#): the **robot in VelObs is able to predict correctly the humans' paths**. In fact, VelObs is the robustest Planning Mode of CoHAN. Here the important note must be done: before letting the robot to navigate in the crowd, the DualBand mode must be manually deactivated because none of the Human Path Prediction services of this mode was designed to produce good results in the crowded environment.

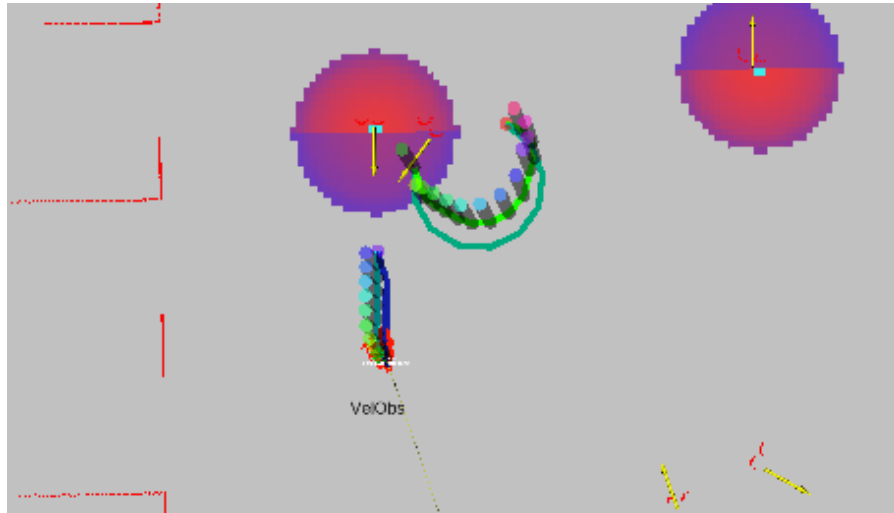


Figure 5.23. Crowded Scenario: Free Space Crowd. The part of Initial Position → Goal 1 path. The robot in VelObs is able to predict the circular trajectory. RViz snapshot.

When considering the path from Goal 1 to Goal 2, the robot continues to navigate in general safely. One of the remarkable examples of the intelligent robot behaviour is encountered here. As it is shown in the [Figure 5.24](#), when a human with a lower velocity emerges on the robot's path, the robot slows down (or stops if necessary) and adapts to the human (velocity and trajectory). Then, it starts following the human behind until its path no longer passes through the risk zone. The questionable social nuance is whether the human feels comfortable with being followed.

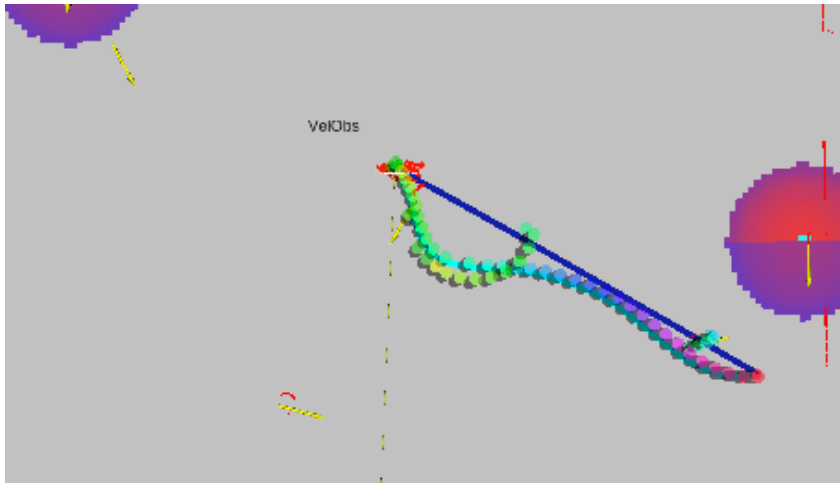


Figure 5.24. Crowded Scenario: Free Space Crowd. The part of Goal 1 \rightarrow Goal 2 path. The robot adapts to the human. RViz snapshot.

Analogously, one particularly noteworthy aspect of the robot behaviour is demonstrated on the path from Goal 2 to Goal 3. As it is illustrated in [Figure 5.25](#), the robot is confronted with a situation in which it is trapped between a static human and inflated static obstacles (beds) on one side and a human moving extremely slowly in a near vicinity on the other side. The slow human temporarily blocks the robot. The robot resolves the temporary blockage by oscillation. While oscillating forward the robot checks if the person has moved away: if not, the robot continues to oscillate, otherwise, it proceeds to the goal.

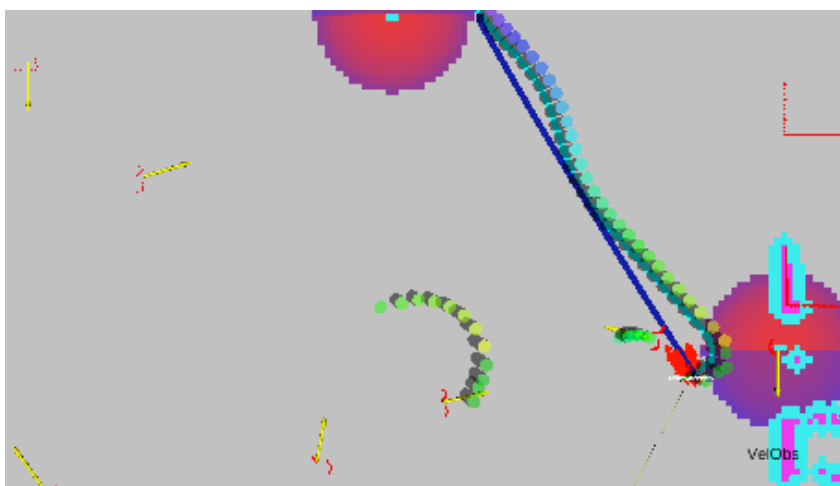


Figure 5.25. Crowded Scenario: Free Space Crowd. The part of Goal 2 \rightarrow Goal 3 path. The temporary blockage resolution by oscillation. RViz snapshot.

Finally, the path from Goal 3 to Goal 4 (which is depicted in the [Figure 5.26](#)) is not associated with some particular event. Rather, it is distinguishable at the moment of the "parking" maneuver because depending on the navigation goal, sometimes robot can get stuck in inflation area.

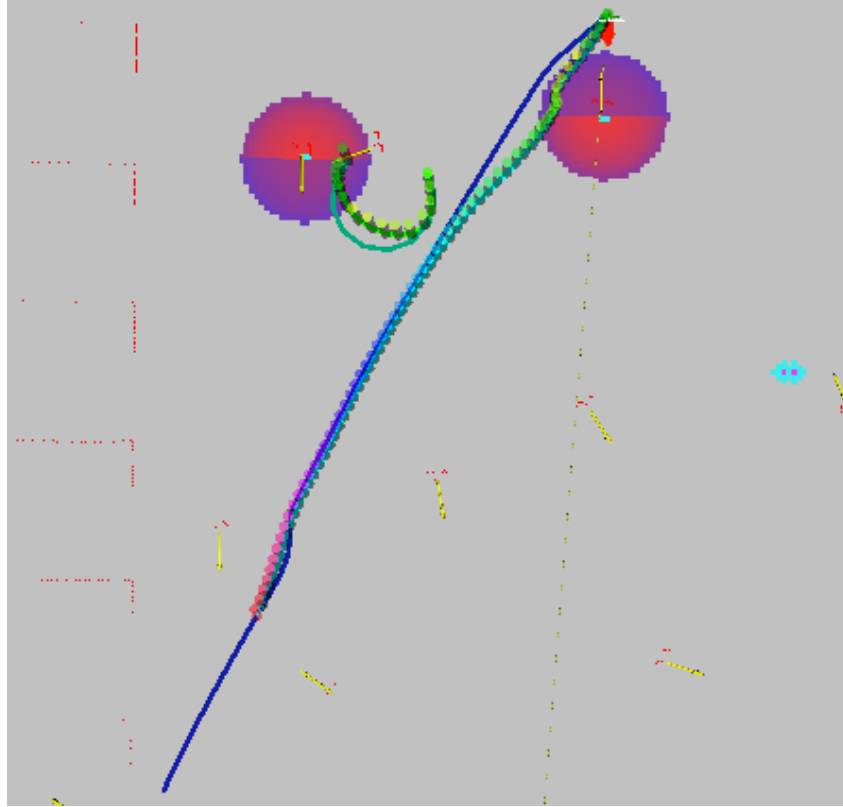


Figure 5.26. Crowded Scenario: Free Space Crowd. The part of Goal 3 → Goal 4 path. RViz snapshot.

The second sub-scenario models the situation of emergency in the ICU – the patient needs an immediate attention or an urgent intervention of the providers, e.g. he/she has a heart attack. In such circumstances the robot's pro-activity is especially important because the robot must not interrupt providers performing the life-saving treatment on a patient. However, the desired robot behaviour is difficult to provide because the robot bases its reasoning on a certain system of beliefs about the state of environment, and this state changes abruptly in emergency.

In the simulation, an emergency is occurring at a specific location: one of the providers is standing next to the bed of the patient with emergency and he/she is calling other providers to help. The scenario assumes that 10 providers suddenly start running to the provider who is calling. They form a "running crowd" next to the bed of the patient with an emergency (Figure 5.27). Meanwhile, the robot had been following a path to some goal (Figure 5.28), but this path now passes through the emergency zone (i.e. comes into conflict with the paths of the "running crowd"). The robot's expected behaviour is to avoid interfering with the crowd.

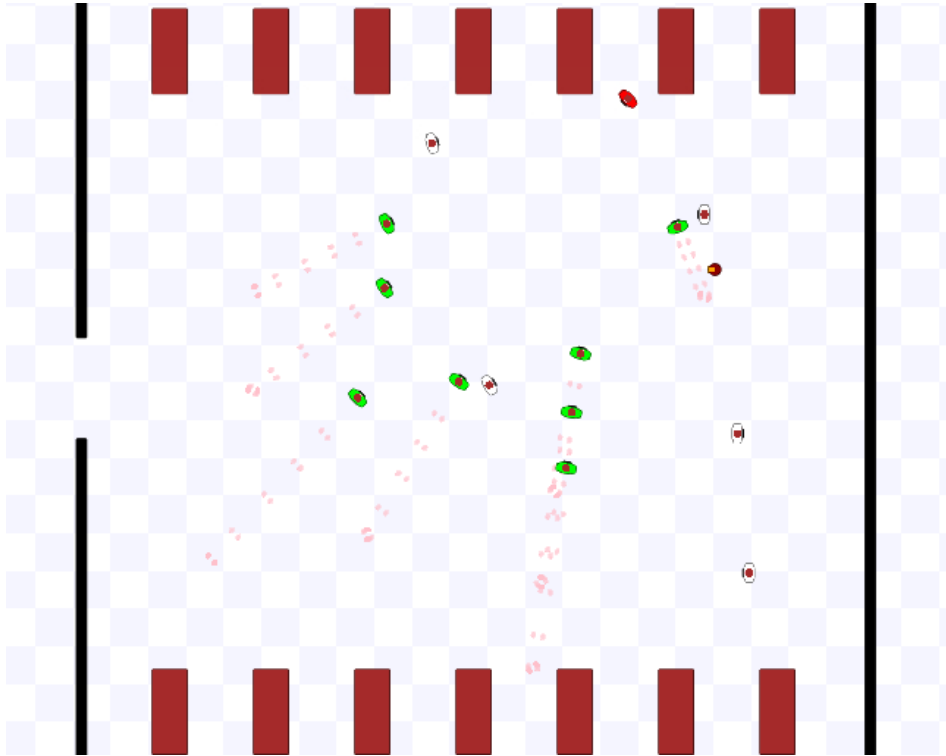


Figure 5.27. Crowded Scenario: Emergency Situation. Initial setup where the humans' motion is represented by their footprints. There are 14 people in the environment, divided into three categories: **1 red human** – "emergency" – location of emergency (e.g. this is the provider who was the closest to the bed of the patient); **8 green humans** – "help" – providers who rush to the location of emergency (each of them performs the manoeuvre consisting of first – correction of the orientation, and next – straight line movement); **5 white humans** – "neutral" – providers who cannot abandon their current tasks and remain at the initial positions in the map. Stage snapshot.

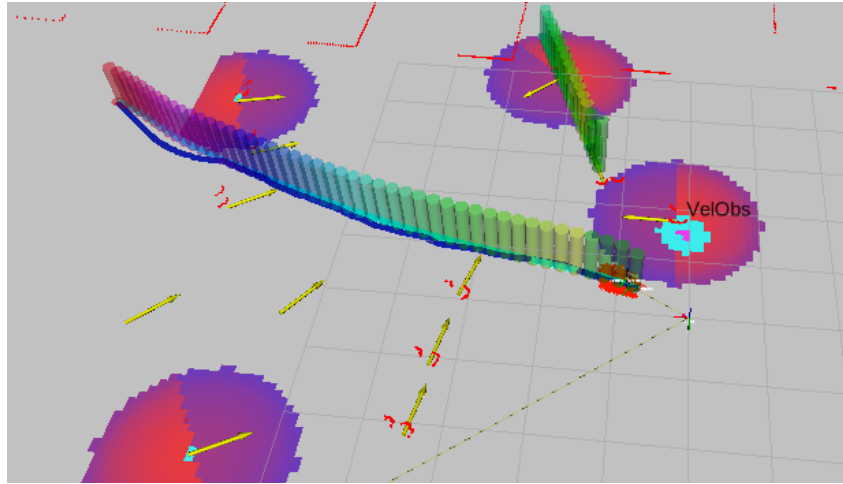


Figure 5.28. Crowded Scenario: Emergency Situation. The robot's global path interferes with the paths of the "running crowd". RViz snapshot.

The `DualBand` mode can be activated as in this case it is not so critical as in the Crowded Scenario: Free Space Crowd. If it is activated, the `PredictGoal` service must be used and configured with the well-defined set of goals. In the analysis below, the `DualBand` is deactivated and the robot operates in `VelObs` whenever dynamic humans are entering the area, limited by the Planning Radius. The robot collision never occurs. However, the human collision occurs depending on the test run.

In lucky runs, the robot is able to re-plan and avoid safely either by passing around the crowd (which is primarily possible due to the fortunate timing), as presented in the [Figure 5.29](#), or by following its path while adapting the speed until the "safe spot" (the spot that is not on the way of any human) is found and halting at this spot to wait for the crowd to move away ([Figure 5.30](#)). When the robot is less lucky, it gets trapped by the humans with no "safe spot" found. In this case a human collision happens ([Figure 5.31](#)): the robot slows down and comes to a halt at the point where it realizes there is no "safe spot", and one or more running humans collide with it. The issue once again uncovers a point for improvement in the **robot's inability to be anticipative enough to foresee the human collision and take a look-ahead action**. Additionally, it is linked to the fact that the robot derives plan predictions for the two nearest dynamic humans only.

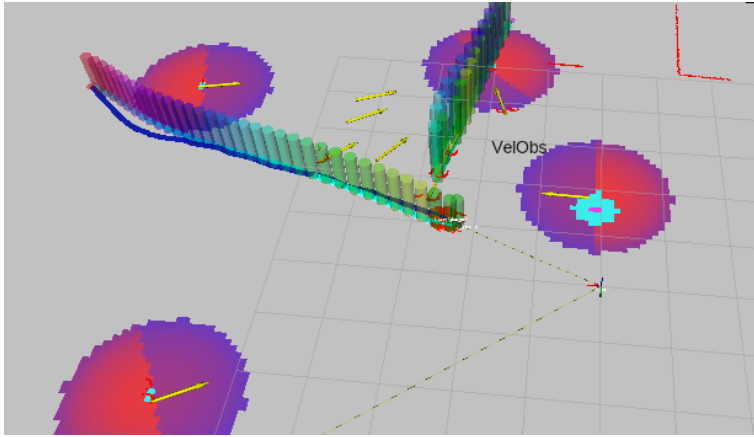


Figure 5.29. Crowded Scenario: Emergency Situation. Lucky run. The robot passes around the "running crowd". RViz snapshot.

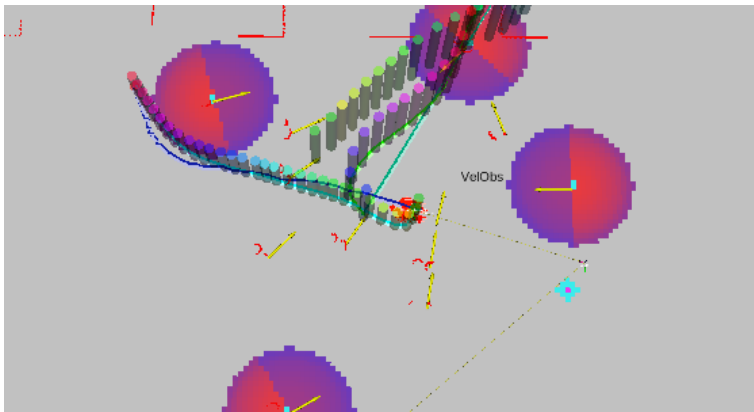


Figure 5.30. Crowded Scenario: Emergency Situation. Lucky run. The robot was able to find a "safe spot". RViz snapshot.

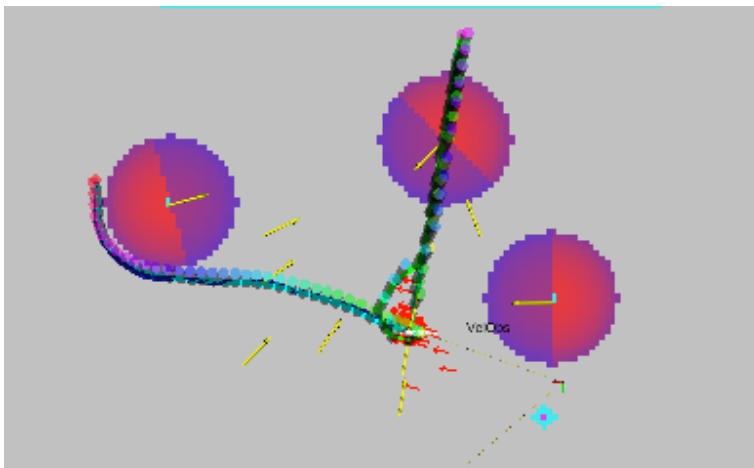


Figure 5.31. Crowded Scenario: Emergency Situation. Unlucky run. No "safe spot" found. RViz snapshot.

5.2.2 Quantitative Results

Five experimental scenarios have been chosen to perform the quantitative analysis, each repeated 10 times with the Co-operative Human-Aware Navigation (CoHAN) and Simple Move Base (SMB) (which exploits a Dijkstra-based function and the Dynamic Window Approach (DWA)) systems in order to compare an integrated Social Navigation planner to a non-human-aware approach such as those found in MARRtino software prior to integration. The averaged over 10 runs results are collected in the [Table 5.1](#).

The tested scenarios are the Door Crossing: Bed Approach, Narrow Corridor: move-and-stop human, Wide Corridor: Free Corridor, Crowded Scenario: Free Space Crowd and Crowded Scenario: Emergency Situation. Only scenarios with dynamic humans are considered in the comparative table. In all these scenarios, both systems produced consistent results over the repetitions. However, in the Door Crossing: Bed Approach and Crowded Scenario: Emergency cases, SMB failed to complete a collision-free navigation.

Table 5.1. Mean values of the metrics over 10 repetitions in five different scenarios: % – Accuracy, PL – Path Length, TT – Total Time, HRD – Minimum Human-Robot Distance (0, if at least 1 of 10 repetitions resulted into collision). The test failure is denoted by "-".

Scenario	CoHAN				SMB			
	%	PL[m]	TT[s]	HRD[m]	%	PL[m]	TT[s]	HRD[m]
<i>Bed Approach</i>	76	5.07	13.04	0.51	-	4.34	-	0
<i>Narrow Corr.</i>	50	31.12	93.99	1.19	100	8.92	34.52	2.98
<i>Wide Corr.</i>	100	16.73	27.66	0.89	100	16.22	32.02	0.50
<i>Fr.Sp.Crowd</i>	72	71.66	155.06	0	50	55.51	138.46	0
<i>Emergency S.</i>	50	16.98	43.78	0	-	9.65	-	0

The metrics involved into comparison are the accuracy, the total length of the path (path length, PL), the total execution time (total time, TT) and the minimum human-robot distance that the planner encountered during navigation (HRD).

The accuracy is the success rate of the test scenario – the percentage of times when test completed successfully. The "success" is defined as reaching the navigation

goal without colliding (both robot and human collisions are counted) and getting stuck, except for the the Narrow Corridor where it is redefined as a detection of the Blockage followed by abortion after which the robot can be tasked with a new navigation goal. This always happens when the robot is controlled with a Simple Move Base, but CoHAN distinguishes between the robot aborting and getting stuck. The navigation cannot be resumed in the latter case unless the Navigation Stack is reloaded, and this occurs 50% of the time. **In all the scenarios, besides the corridors, CoHAN outperforms SMB in accuracy.** In fact, SMB does not avoid human collisions at all in the Bed Approach and Emergency Situation scenarios, whereas using a new planner, the robot sometimes gets lost in the bed inflation or is not able to find a "safe spot", but sometimes resolves the conflict. In the Free Space Crowd, the robot with SMB can only navigate safely when it is lucky. The CoHAN system, on the other hand, produces rare collisions due to lag in costmap updates during mode switching. Finally, both approaches are reliable in the free wide corridors although although the SMB's local plans are built reactively while CoHAN exhibits proactive behaviours.

The path length is the total length of the path computed as the sum of the distances between every sequential pair of states (robot's ground truth positions). When SMB tests fail, the length of the generated global path is computed instead because SMB-driven robots rarely deviate significantly from this path. The [Table 5.1](#) shows that in all test instances **SMB made robot travel shorter distances.** This is explained by HATEB using larger deviations for early intention show (due to the relative velocity constraint in the optimisation scheme), proactive elastic bands and driving away maneuvers. Nonetheless, there is a specific reason for such a large difference in path lengths in case of the Narrow Corridor scenario – the non-socially navigating robot comes to a halt as soon as it detects a blocking obstacle, and the CoHAN-based system, before declaring abortion or understanding that the robot is stuck, spends some time on a mode switching and subsequent re-planning while oscillating. The same reason explains why CoHAN takes nearly three times as

long as SMB to complete the navigation in a given scenario and why the minimum human-robot distance is more than two times shorter.

Due to proactive deviation maneuvers, performed by CoHAN, the total execution time, taken by it, is greater than the one of SMB unless the human-robot co-navigation context is such that the reactive planner takes longer because the robot needs to slow down in the human vicinity for collision avoidance, as it happens in Wide Corridor experiments. The same velocity and acceleration limits of the robot are provided to both the planners.

Finally, when it comes to the minimum human-to-robot distances, the HATEB's behaviour varies because it performs the situation assessment and addresses each case individually. **If the space is available, the integrated planner keeps a greater minimum distance from the human than SMB.** Otherwise, it can also choose the strategy of slowing down and approaching closer.

In summary of the quantitative evaluation, the integrated system proves its human-awareness and can be considered outperforming in all simulated scenarios except for the Narrow Corridor where the Simple Move Base appears to exhibit more reasonable and less resource-consuming behaviour. However, even in the latter case the conceptual difference is not large and can be mitigated by debugging the `Backoff-recovery` mode.

5.2.3 Human Evaluation Results

In order to assess human acceptance of the assistive robot driven by the integrated system and estimate its usability in health-care facilities, 10 video demonstrations of the robot's behaviour were presented to the physicians at Sant'Andrea Hospital:

1. **Door Crossing: Wide Doorway.** Simultaneous crossing by the robot and the human. The human is assumed cooperative. [Figure 5.7](#);

2. **Door Crossing: Narrow Doorway.** A human-robot crossing. The human is assumed cooperative, and the robot moves backwards. [Figure 5.8](#);
3. **Door Crossing: Bed Approach.** When the robot detects a human, it drives to the side and waits. [Figure 5.10](#);
4. **Wide Corridor: Free Corridor.** A perfect run. The robot takes more effort and passes the human by. [Figure 5.16](#);
5. **Wide Corridor: Cluttered Corridor.** When the robot detects a human, it starts moving backwards to let the human pass. After a while of moving backwards, the robot stops. [Figure 5.20](#);
6. **Narrow Corridor: Human who moves, but then stops.** The robot, blocked by the human, switches to **Backoff-recovery**, oscillates and stops. [Figure 5.13](#);
7. **Crowded Scenario: Free Space Crowd-1.** A part of Goal 1 \rightarrow Goal 2 path where the robot adapts to the human. [Figure 5.24](#);
8. **Crowded Scenario: Free Space Crowd-2.** A part of Goal 2 \rightarrow Goal 3 path. The temporary blockage resolution by oscillation. [Figure 5.25](#);
9. **Crowded Scenario: Emergency Situation-1.** A lucky run where the robot was able to find a "safe spot". [Figure 5.30](#);
10. **Crowded Scenario: Emergency Situation-2.** An unlucky run with no "safe spot" found. [Figure 5.31](#).

The recipients we asked to rate how satisfied they were with the robot's behaviour in each demonstrated scenario on a 5-point grading scale, with 1 representing absolutely unacceptable behaviour and 5 representing perfection. The feedback form was filled out by 40 participants, and their average acceptance results are reported in aggregated form in [Table 5.2](#) and in a profiled by scenarios form in [Table 5.3](#).

The [Table 5.2](#) shows that the **average level of the robot acceptability is high and can be rounded up to 82%, regardless of the nature or difficulty of the simulated scenario**. Thus, domain experts were not predisposed to higher cautiousness within more domain-specific or chaotic settings, resulting in a positive outlook on the robot’s future integration in a safety-critical environment.

Table 5.2. Average robot acceptability: Avg – in entire population, Avg Med – in medical contexts (scenarios 3,5,7-10), Avg Non-Med – in common sense contexts (scenarios 1,2,4,6), Avg Crowded – in crowded environments (scenarios 7-10), Avg Non-Crowd – in environments with 1 dynamic human (scenarios 1-6).

Metric	Value[%]
Avg	81.7
Avg Med	81.5
Avg Non-Med	81.9
Avg Crowd	81.5
Avg Non-Crowd	81.8

Furthermore, [Table 5.3](#) confirms the hypothesis of clustered, scenario-independent results, as none of the average values differ significantly from the others.

Table 5.3. Average robot acceptability and the standard deviation from the average estimate computed for each scenario.

Scenario №	1	2	3	4	5	6	7	8	9	10
Avg Value [%]	80.5	82.6	82.1	82.6	81.0	82.1	82.1	82.1	80.5	81.5
STD	0.74	0.73	0.82	0.73	0.83	0.75	0.68	0.68	0.71	0.58

It is worth noting that the lowest encountered estimate in all demonstrated scenarios was 3, indicating that **nobody of the participating providers rated the robot’s behaviour as more likely to be unacceptable than acceptable**. And another indicator of consistency is the median value, which is 4 in all cases and is equal to the mode value in all cases, except for the Door Crossing: Bed Approach and Wide Corridor: Cluttered Corridor where the mode is 5. To gain more intuition on the distribution of the grades assigned by experts to each video simulation let us expand [Table 5.3](#) into a bar chart in [Figure 5.32](#) that computes percentages of estimates of each type.

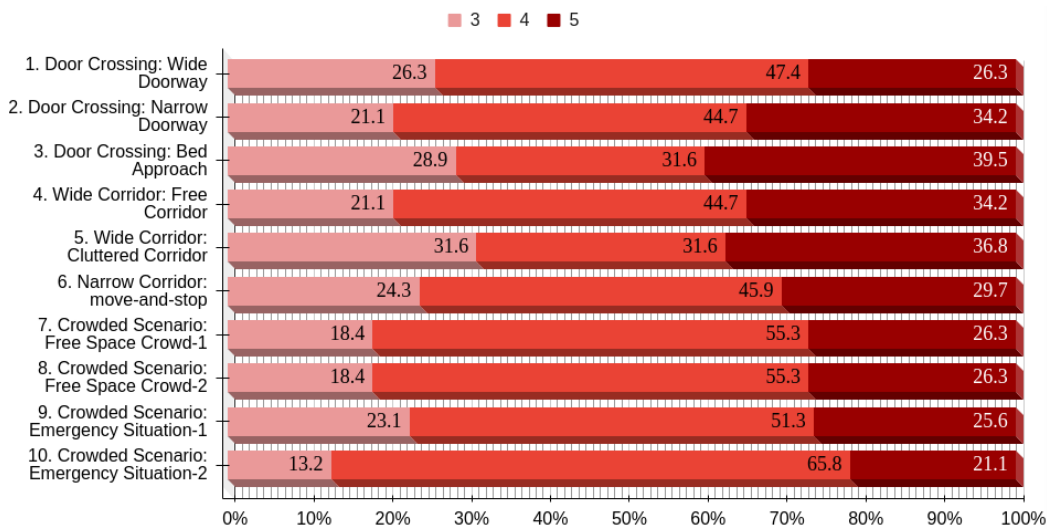


Figure 5.32. The percentages distribution of grades in each scenario presented to health-care workers.

A precise examination of the chart leads to the conclusion that **the robot's behaviour cannot be considered more acceptable in some cases than in other**. For example, two mentioned previously scenarios, Door Crossing: Bed Approach and Wide Corridor: Cluttered Corridor, have the highest concentration of both minimum and maximum given estimates, 3 and 5, that is coherent to higher standard deviations in these cases, seen from [Table 5.3](#), but does not allow to mark these scenarios as neither preferred nor disregarded by clinicians.

Overall, it seems from the chart in [Figure 5.32](#) and from the calculated statistics in general that even confusing and causing inconveniences for humans robot behaviours such as following behind the person (Crowded Scenario: Free Space Crowd-1), stopping at random in the middle of the crowd (Crowded Scenario: Emergency Situation-2) or blocking the human's way (Narrow Corridor: move-and-stop human) would be tolerated by people. As a result, **expert validation of the robot's performance brings to a positive conclusion about possibility of human-robot coexistence in health-care facilities.**

Chapter 6

Conclusion and Future Work

The Human-Aware Navigation is a field that broadens the horizon of the robotic research by empowering existing robotic systems with socially meaningful capabilities and contributing to the seamless integration and sustainability of the robots in the human environments. It proposes a wide range of challenging tasks, associated with incorporation of the socially enhanced components into established navigation frameworks and methodologies. One of such tasks is an enhancement of the robot reliability in health-care sector by means of Human-Aware Navigation Planning that has been addressed in this thesis. In our case, the socially enhanced component is a Co-operative Human-Aware Navigation (CoHAN) planner and the established navigation framework is a MARRtino robot software.

First of all, **the results of the comparison with a non-social planning approach manifest an improvement in robot reliability**, and the qualitative analysis demonstrates that the robot behaviour became more socially compliant which promotes human acceptance of the robot, as confirmed by statistical analysis of human validation of the results.

Furthermore, the CoHAN is the first Human-Aware Navigation component integrated into MARRtino. This suggests the scope of future investigations for the community of people who are interested in using this robot for their studies and

development. **The integration of CoHAN provides a baseline** for comparison and a convenient starting point for the aforementioned investigations.

Then, probably, **the greatest value of this work consists in a thorough examination** of the planner’s features. While acknowledging the limitations of the system’s potential applications, it also provides an intuition on the directions of the future system-upgrading initiatives.

A room for improvement has been found in the **resilience of the Backoff-recovery** method and further development of its prototypical edition. Then, **the goal selection in the PredictGoal** service can be automatised by implementation of some probabilistic goal inference approach. Besides, **new Human Path Prediction techniques can be embedded** into system to meet a need for improved estimation of human motion that has been verified throughout experimentation. The experimentation also suggests that **the human model has to be updated**. Eventually, **the proactive and anticipative behaviour on the side of the robot are two particularly important aspects**, and they are even more critical in medical settings. Therefore, inclusion of new methods with a design focus on strengthening the system in these aspects would be beneficial.

Shading more light on vectors of the system upgrades, we plan a **re-integration of a current version of the planner with a newer one** [38]. The key motivation for doing so is in boosting up the robot capability to act proactively by accounting also for humans outside the Visible Region. Consequently, another series of experiments would need to be conducted using a newly integrated version of CoHAN in order to attest the issues found in the outdated version.

In addition, the system improvement challenge can be approached from a completely different angle: **the human controller module can be enhanced**, for instance, by creating human avatars that demonstrate rational social behaviours rather than just moving in a primitive way according to predefined rules. The insight-

ful works on that are presented by Favier et. al in [13] and [19]. The authors propose an Intelligent Human Simulation (InHuS) system that incorporates autonomous intelligent human agents, specifically designed to act and interact with a robot navigating in a simulated environment. Since this system is based on CoHAN, it would be interesting to combine it with already integrated planner in the MARRtino software.

Finally, in order to complete a characterisation of the system, **we intend to deploy it on a physical robotic platform and introduce to a real-world clinical environment**, as the estimated level of the robot acceptability by experts of application domain encourages such introduction. Then, during the process of deployment, new challenges will likely be identified, necessitating the development of new solutions. After these solutions are found, **extensions to non-medical contexts and other robots** can be investigated.

Appendix

A Final System Configuration

Table a. Parameters of move_base.

Parameter	Value
planner_frequency	8.0
controller_frequency	8.0
planner_patience	10.0
controller_patience	15.0
recovery_behaviour_enabled	true
max_planning_retries	3
oscillation_timeout	15.0
oscillation_distance	0.3

Table b. Global/Local Costmap plugins.

Name	Type
static	costmap_2d::StaticLayer
obstacles	costmap_2d::ObstacleLayer
inflater	costmap_2d::InflationLayer
human_layer_static	human_layers::StaticHumanLayer
human_layer_visible	human_layers::HumanVisibilityLayer

Table c. Global/Local Costmap inflaters.

Costmap	Inflation Radius
global	0.4
local	0.2

Table d. Parameters of the Robot section of HATEB Local Planner.

Parameter	Value
holonomic_robot	false
max_vel_y	0.0
acc_lim_y	0.0
max_vel_x	0.7
min_vel_x	0.02
max_vel_x_backwards	0.4
max_vel_theta	1.2
min_vel_theta	0.1
acc_lim_x	0.3
acc_lim_theta	0.4
min_turning_radius	0.0

Table e. Parameters of the Obstacles section of HATEB Local Planner.

Parameter	Value
min_obstacle_dist	0.2
include_costmap_obstacles	true
costmap_obstacles_behind_robot_dist	0.5
obstacle_poses_affected	1
costmap_converter_spin_thread	true
costmap_converter_rate	10
obstacle_cost_mult	1.0
use_nonlinear_obstacle_penalty	true
cluster_max_distance	0.8
cluster_min_pts	2
ransac_inlier_distance	0.15
ransac_min_inliers	10
ransac_no_iterations	1500
ransac_remaining_outliers	3
ransac_convert_outlier_pts	true
ransac_filter_remaining_outlier_pts	false
convex_hull_min_pt_separation	0.1

Table f. Parameters of the Human Model section of HATEB Local Planner. `false/true` means that value of the parameter varies depending on the context.

Parameter	Value
human_radius	0.3
min_human_dist	0.4
max_human_vel_x	1.3
max_human_vel_y	0.0
nominal_human_vel_x	1.1
max_human_vel_x_backwards	0.01
max_human_vel_theta	1.1
human_acc_lim_x	0.6
human_acc_lim_y	0.0
human_acc_lim_theta	0.8
use_external_prediction	false/true
predict_human_behind_robot	false/true
predict_human_goal	false/true
human_robot_ttc_scale_alpha	1.0
human_robot_ttcplus_scale_alpha	1.0
human_min_samples	3
use_human_robot_safety_c	true
use_human_robot_rel_vel_c	true
use_human_robot_visi_c	true
use_human_human_safety_c	true
use_human_robot_ttc_c	false
use_human_robot_ttcclosest_c	false
use_human_robot_ttcplus_c	false
scale_human_robot_ttc_c	false
scale_human_robot_ttcplus_c	false
ttc_threshold	5.0
ttclosest_threshold	1.0
ttcplus_threshold	5.0
ttcplus_timer	1.0
rel_vel_cost_threshold	1.5
visibility_cost_threshold	2.5
min_human_robot_dist	0.6

Table g. Parameters of the Optimisation section of HATEB Local Planner.

Parameter	Value
no_inner_iterations	8
no_outer_iterations	4
optimization_activate	true
optimization_verbose	false
penalty_epsilon	0.01
weight_max_vel_x	2
weight_max_vel_y	2
weight_max_human_vel_x	4
weight_max_human_vel_y	4
weight_nominal_human_vel_x	2
weight_max_vel_theta	1
weight_max_human_vel_theta	2
weight_acc_lim_x	1
weight_acc_lim_y	1
weight_human_acc_lim_x	2
weight_human_acc_lim_y	2
weight_acc_lim_theta	1
weight_human_acc_lim_theta	2
weight_kinematics_nh	1.0
weight_kinematics_forward_drive	1.0
weight_kinematics_turning_radius	0
weight_optimaltime	1.0
weight_human_optimaltime	3.0
weight_obstacle	50.0
weight_dynamic_obstacle	50
weight_viapoint	0.05
weight_human_viapoint	0.5
weight_human_robot_safety	2.0
weight_human_human_safety	2.0
weight_human_robot_ttc	1.0
weight_human_robot_rel_vel	5.0
weight_human_robot_ttcplus	1.0
weight_human_robot_visibility	5.0
weight_shortest_path	0
selection_alternative_time_cost	false
cap_optimaltime_penalty	true

Table h. Parameters of the Trajectory section of HATEB Local Planner.

Parameter	Value
teb_autosize	true
dt_ref	0.3
dt_hysteresis	0.1
global_plan_overwrite_orientation	true
allow_init_with_backwards_motion	false
max_global_plan_lookahead_dist	0.0
feasibility_check_no_poses	2
global_plan_viapoint_sep	0.2
disable_warm_start	true
shrink_horizon_backup	true

Table i. Parameters of the Goal Tolerance section of HATEB Local Planner.

Parameter	Value
xy_goal_tolerance	0.4
yaw_goal_tolerance	0.2
free_goal_vel	true

Table j. Parameters of the Homotopy Class section of HATEB Local Planner.

Parameter	Value
enable_homotopy_class_planning	false
enable_multithreading	true
simple_exploration	true
max_number_classes	4
roadmap_graph_no_samples	15
roadmap_graph_area_width	5
h_signature_prescaler	0.5
h_signature_threshold	0.1
obstacle_keypoint_offset	0.1
obstacle_heading_threshold	0.45
visualize_hc_graph	true

Bibliography

- [1] Ros concepts. <http://wiki.ros.org/ROS/Concepts>.
- [2] Ros navigation. <http://wiki.ros.org/navigation>.
- [3] Ros stage. <http://wiki.ros.org/stage>.
- [4] Stanford artificial intelligence robot. <http://stair.stanford.edu/>.
- [5] Timed elastic band. http://wiki.ros.org/teb_local_planner.
- [6] Rachid Alami, K Madhava Krishna, and Thierry Siméon. Provably safe motions strategies for mobile robots in dynamic domains. *Autonomous Navigation in Dynamic Environments of Springer Tracts in Advanced Robotics*, 35:85–106, 2007.
- [7] Santosh Balajee Banisetty, Scott Forer, Logan Yliniemi, Monica Nicolescu, and David Feil-Seifer. Socially-aware navigation: A non-linear multi-objective optimization approach. *arXiv preprint arXiv:1911.04037*, 2019.
- [8] Santosh Balajee Banisetty, Vineeth Rajamohan, Fausto Vega, and David Feil-Seifer. A deep learning approach to multi-context socially-aware navigation. *IEEE/RSJ IROS*, 2020.
- [9] Subhrajit Bhattacharya, Vijay Kumar, and Maxim Likhachev. Search-based path planning with homotopy class constraints. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

- [10] Alejandro Bordallo, Fabio Previtali, Nantas Nardelli, and Subramanian Ramamoorthy. Counterfactual reasoning about intent for interactive navigation in dynamic environments. *IEEE/RSJ IROS*, 2015.
- [11] Manuel Fernandez Carmona, Tejas Parekh, and Marc Hanheide. Making the case for human-aware navigation in warehouses. *Annual Conference Towards Autonomous Robotic Systems, Springer*, 2019.
- [12] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. *IEEE/RSJ IROS*, 2017.
- [13] Anthony Favier, Phani Teja Singamaneni, and Rachid Alami. Challenging human-aware robot navigation with an intelligent human simulation system. <https://hal.laas.fr/hal-03684245>, 2022.
- [14] Gonzalo Ferrer, Anaís Garrell, and Alberto Sanfeliu. Social-aware robot navigation in urban environments. *IEEE ECMR*, 2013.
- [15] Gonzalo Ferrer and Alberto Sanfeliu. Multi-objective cost-to-go functions on robot navigation in dynamic environments. *IEEE/RSJ IROS*, 2015.
- [16] Gonzalo Ferrer and Alberton Sanfeliu. Bayesian human motion intentionality prediction in urban environments. *Pattern Recognition Letters*, 2014.
- [17] Jaime F Fisac, Andrea Bajcsy, et al. Probabilistically safe robot planning with confidence-based human predictions. *arXiv preprint arXiv:1806.00109*, 2018.
- [18] Ronja Guldenring, Michael Gerner, Norman Hendrich, Niels Jul Jacobsen, and Jianwei Zhang. Learning local planners for human-aware navigation in indoor environments. *IEEE/RSJ IROS*, 2020.
- [19] Olivier Hauterville, Camino Fernández, Phani Teja Singamaneni, Anthony Favier, Vicente Matellán, and Rachid Alami. Imhus: Intelligent multi-human simulator. *IROS2022 Workshop: Artificial Intelligence for Social Robots Interacting with Humans in the Real World*, 2022.

- [20] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.
- [21] Harmish Khambhaita and Rachid Alami. Viewing robot navigation in human environment as a cooperative activity. *International Symposium on Robotics Research*, 2017.
- [22] Marina Kollmitz, Kaijen Hsiao, Johannes Gaa, and Wolfram Burgard. Time dependent planning on a layered social cost map for human-aware robot navigation. *IEEE ECMR*, 2015.
- [23] K Madhava Krishna, Rachid Alami, and Thierry Siméon. Moving safely but not slowly - reactively adapting paths for better trajectory times. *11th International Conference on Advanced Robotics*, 2003.
- [24] K Madhava Krishna, Rachid Alami, and Thierry Siméon. Safe proactive plans and their execution. *Robotics and Autonomous Systems*, 54:244–255, 2006.
- [25] Thibault Kruse, Amit K Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 2013.
- [26] David V Lu, Dave Hershberger, and William D Smart. Layered costmaps for context-sensitive navigation. *IEEE/RSJ IROS*, 2014.
- [27] Dhanvin Mehta, Gonzalo Ferrer, and Edwin Olson. Autonomous navigation in dynamic social environments using multi-policy decision making. *IEEE/RSJ IROS*, 2016.
- [28] Billy Okal and Kai O Arras. Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning. *IEEE ICRA*, 2016.
- [29] Noé Pérez-Higueras, Fernando Caballero, and Luis Merino. Teaching robot navigation behaviors to optimal rrt planners. *International Journal of Social Robotics*, 2018.

- [30] Noé Pérez-Higueras, Rafael Ramón-Vigo, Fernando Caballero, and Luis Merino. Robot local navigation with learned social cost functions. *11th International Conference on Informatics in Control, Automation and Robotics*, 2014.
- [31] Kun Qian, Xudong Ma, Xianzhong Dai, Fang Fang, and Bo Zhou. Decision-theoretical navigation of service robots using pomdps with human-robot co-occurrence prediction. *International Journal of Advanced Robotic Systems*, 2013.
- [32] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. *ICRA workshop on open source software*, 3, 2009.
- [33] Ely Repiso, Gonzalo Ferrer, and Alberto Sanfeliu. On-line adaptive side-by-side human robot companion in dynamic urban environments. *IEEE/RSJ IROS*, 2017.
- [34] Christoph Roesmann, Wendelin Feiten, Thomas Woesch, Frank Hoffmann, and Torsten Bertram. Trajectory modification considering dynamic constraints of autonomous robots. *ROBOTIK 2012; 7th German Conference on Robotics*, pages 74–79, 2012.
- [35] Christoph Rosmann, Frank Hoffmann, and Torsten Bertram. Planning of multiple robot trajectories in distinctive topologies. *IEEE European Conference on Mobile Robots*, 2015.
- [36] Phani Teja Singamaneni and Rachid Alami. Hateb-2: Reactive planning and decision making in human-robot co-navigation. *International Conference on Robot & Human Interactive Communication*, 2020.
- [37] Phani Teja Singamaneni, Anthony Favier, and Rachid Alami. Human-aware navigation planner for diverse human-robot interaction contexts. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

- [38] Phani Teja Singamaneni, Anthony Favier, and Rachid Alami. Watch out! there may be a human. addressing invisible humans in social navigation. *arXiv:2211.12216*, 2022.
- [39] Emrah Akin Sisbot, Luis F Marin-Urias, Rachid Alami, and Thierry Simeon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 2003.
- [40] Angelique M Taylor, Sachiko Matsumoto, and Laurel D Riek. Situating robots in the emergency department. *AAAI Spring Symposium on Applied AI in Healthcare: Safety, Community, and the Environment*, 2020.
- [41] Angelique M Taylor, Sachiko Matsumoto, Wesley Xiao, and Laurel D Riek. Social navigation for mobile robots in the emergency department. *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [42] Rudolph Triebel, Kai O Arras, Rachid Alami, et al. Spencer: A socially aware service robot for passenger guidance and help in busy airports. *Field and service robotics, Springer*, 2016.
- [43] Xuan-Tung Truong and Trung-Dung Ngo. Toward socially aware robot navigation in dynamic and crowded environments: A proactive social motion model. *IEEE Transactions on Automation Science and Engineering*, 2017.
- [44] Xuan-Tung Truong, Voo Nyuk Yoong, and Trung-Dung Ngo. Dynamic social zone for human safety in human-robot shared workspaces. *11th International Conference on Ubiquitous Robots and Ambient Intelligence*, 2014.
- [45] Keenan A Wyrobek, Eric H Berger, Machiel H.F Van der Loos, and Kenneth J Salisbury. Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. *IEEE International Conference on Robotics and Automation*, 2008.