Nav2CAN

[ˈnæv ˈtuˌkæn]

Nav2 Context Aware Navigation

ROB 10

ROB 1056

Robotics
Aalborg University

AALBORG UNIVERSITY

STUDENT REPORT

Robotics
Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg
http://www.robotics.aau.dk

Title:
Nav2CAN

Theme:
Context Aware mobile robotics

Project Period:
Spring 2023

Project Group:
ROB 1056

Participant(s):
Jonathan E. Schmidt
Tristan Schwörer

Supervisor(s):
Dimitris Chrysostomou

Copies: 0

Page Count: 77

Date of Completion:
June 1, 2023

Abstract:

This thesis presents Nav2CAN, a modular system that can be added to mobile robots running ROS2 to achieve context aware navigation in human filled environments. Nav2CAN enables the implementation of costmap plugins that adds social proxemics as well as interaction proxemics when multiple people are standing in F-formations. This means that any Nav2 stack, using local or global costmaps, can be expanded to achieve context aware navigation. The project also implements a method for people detection locally on the robot but is suited for using other internal or external methods for detecting humans. Through simulation and module tests in the real world Nav2CAN proves that this approach for context aware navigation enables a human aware navigation that can potentially increase the comfort of humans in the proximity of the navigating robot. The code for Nav2CAN is provided at `https://github.com/Tristan9497/master-thesis`.

Preface

This master thesis signifies the work during the semester of spring 2023 as well as being the culmination of our master studies in Robotics at Aalborg University. The work presented in this thesis also marks the beginning of our future within robotics which is filled with curiosity for what this rapidly expanding field will achieve and how we might impact it.

We have been fortunate to delve into different aspects within robotics, especially mobile robotics, and have found a shared passion for trying to understand the intricacies that is context aware navigation. Especially, understanding how humans and robots can collaborate and how to design for human comfort while perceiving human behaviour and preferences has been a guideline for the work developed during our studies.

We appreciate the guidance and support we have received during our studies from staff at AAU and especially from Associate Professor Dimitris Chrysostomou who has guided us through multiple projects with expertise and encouragement.

Aalborg University, June, 2023

Jonathan Eichild Schmidt
<jschmi17@student.aau.dk>

Tristan Schwörer
<tschwo21@student.aau.dk>

Acronyms and abbreviations

AI	Artificial Intelligence
AMR	Autonomous Mobile Robot
AP	Average Precision
CNN	Convolutional Neural Network
DBN	Dynamic Bayesian Network
DSG	Dynamic Scene Graphs
E-ELAN	Extended Efficient Layer Aggregation Network
FOV	Field of View
GPU	Graphics Processing Unit
HHI	Human-Human Interaction
HOI	Human-Object Interaction
HRI	Human-Robot Interaction
IR	Infrared
IU	Interaction Units
KF	Kalman Filter
Nav2	Navigation 2
Nav2CAN	Nav2 Context Aware Navigation
NLP	Natural Language Processing
QoS	Quality of Service
RGB	Red-Green-Blue
ROS	Robot Operating System
ROS1	Robot Operating System 1
ROS2	Robot Operating System 2
RRT	Rapidly-exploring Random Tree
SLAM	Simultaneous Localization And Mapping
TBoF	Trainable Bag of Freebies
VR	Virtual Reality

Contents

1 - Introduction

Context is defined by Merriam-Webster's dictionary [1] as: *the interrelated conditions in which something exists or occurs.* Therefore, context aware navigation means to navigate while understanding the environment and the actors and objects within that environment.

Mavrogiannis et al. [2] describes navigation as the task of following a collision-free route in an efficient manner from initial position to a goal. This means that calculating and following a route while deviating to avoid collisions is the core concept. When adding social or context aware navigation, the goal is to avoid the obstacles, that is socially uncomfortable zones, from a human perspective. This means that to efficiently be aware of social contexts the mobile robot must be able to calculate the route that will adhere to social norms such that humans will be comfortable around the robot. Therefore, using the current actions of humans to plan paths that will try to avoid interruption is desired.

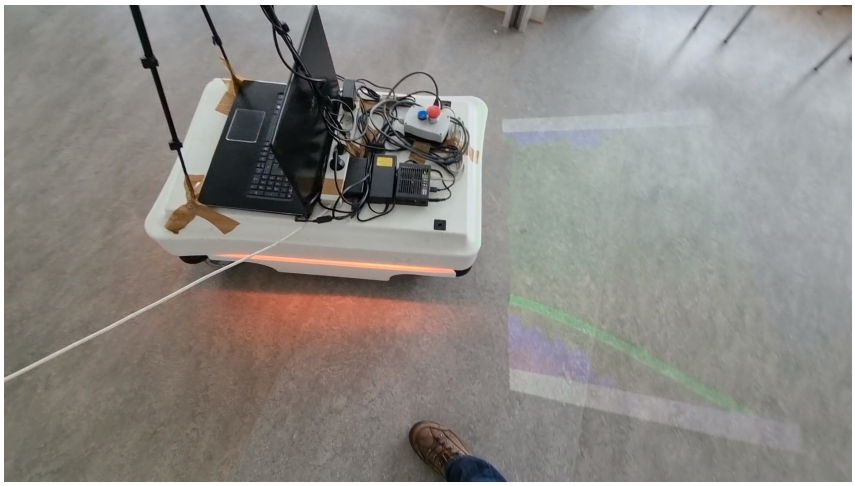


Figure 1.1: A projector used to show the planned path of the robot on the floor. This is used to highlight the intended action of the robot to the user.

A previous project [3] by the authors of this work showed the importance of using social aware navigation when deploying mobile robots in a human filled environment. In that work, the focus was on integrating social aware navigation on a mobile robot platform, that could use the location of a single human to ensure that the human felt safe, comfortable and trusted the robot. The importance of a

robust detection of humans along with a socially aware navigation would improve how the human felt around the robot. Implementing an indication system, as the one seen in figure 1.1, for the robot was part of the previous work, which showed the importance of the communication between robot and human. However, it was evident, that social aware navigation had its limitation due to implementation.

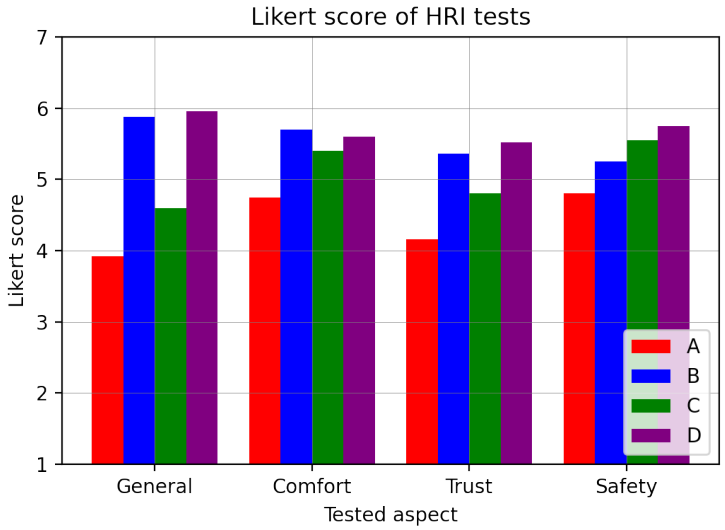


Figure 1.2: The Likert rating of different aspects of the system developed in a previous work. Here A is a test using only the navigation from the mobile platform. B incorporates the indication of the path without social aware navigation. C uses socially aware navigation without indication. D uses both socially aware navigation and indication.

Figure 1.2 shows the different ratings of the system using different setups. Here it was found that indication gave the larger improvement due to the expected movement from the robot. Part of the reason for lower performance was due to implementation, where the robot would move in an oscillatory manner. This showed a need for a better implementation of robotic planning.

To achieve better planning it is important to understand how the planning can take multiple aspects of the environment into consideration. Here both social zones of humans as well as potential interactions between humans are of interest. This means that the robot should be able to include areas of interaction when planning the optimal path in a pursuit to not disturb people. As described by the participants of the previous work, it is important for a robot to behave predictably and consistently for humans to act around it.
This leads to the initial problem formulation that guides the problem analysis of this thesis:

How can a mobile platform achieve context awareness to perform navigation in human filled environments?

2 - Problem Analysis

To understand the needs within social interactions between humans and robots it is important to understand the state-of-the-art research within the area as well as connected fields that might influence the success of implementing context aware navigation. Therefore, this chapter highlights different research within social robotics, Human-Robot Interaction (HRI), context perception, and mobile robot navigation and mapping.

2.1 Context aware navigation

When navigating in an environment that contains dynamic objects and, in particular, humans, it is important for a robot to be able to assess something about a given context. These contexts can vary but generally include humans in one way or another. As described by Lemaignan et al. [4], Artificial Intelligence (AI) must be able to extract knowledge and create models of humans in a context to make decisions that can be translated into actions. Often this knowledge of context is gathered through multiple modalities which further require multiple methods for perception and fusion of sensor data. For mobile robots, the robot must be aware of social rules that humans act according to in order to be perceived as safe and comfortable by the human.

Lemaignan et al. [4] describes the HRI as consisting of three challenges: Communication, Joint action, and Human-Aware execution. Here the communication can be in different modalities, such as speech or gestures. The joint action can be handling an object in collaboration or moving safely and efficiently around a work area. Human-Aware execution relies on the robot's beliefs on the human actions to achieve execution that is not interfering with the human action. Marques et al. [5] proposes a proxemics-based approach for handling activity in the cost map of the robot. This includes the detection of a human interacting with a given object and increasing the affordance space.

Clavero et al. [6], uses adaptive proxemics to determine a social layer for the costmap that changes depending on the context. This adaption is implemented by changing the social zone to be either restricted or a collaboration zone, depending on the context.

2.1.1 Proxemics

As described above, proxemics can be an important factor when developing context aware navigation. Therefore, a brief introduction to the concept of proxemics and uses within robotics are described in the following.

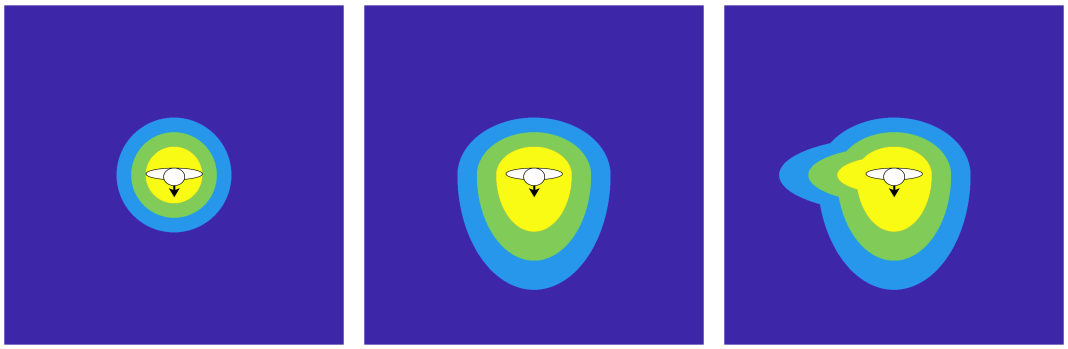
The first definition of proxemics was described by Edward T. Hall [7] as: *the interrelated observations and theories of humans use of space as a specialised elaboration of culture*. This was a description of how humans use space around them and how they perceive other people in the surrounding space. According to Hall, four zones around a human can be described:

1. **Intimate zone** within 0.5m of the person. This zone is reserved for close personal relationship where physical contact is possible if allowed by the person. Invasion of the intimate zone can be considered as a threat and cause discomfort for the person. Certain situations may limit the size of the intimate zone due to spacial limitations.

2. **Personal zone** from .5m to 1.4m from the person. The personal zone a natural interaction can happen with other people where physical contact is possible.

3. **Social zone** from 1.4m to 4m from the person. Here social interactions can happen while keeping physical distance.

4. **Public zone** more than 4m from the person. In this zone attention is not on other people and their identity is unknown. Here interaction is not expected due to the larger distance between people.



(a) Uniform social space **(b)** Social space **(c)** Social space with dominant side

Figure 2.1: Different social space representations based on proxemics theory. Proxemic zones are not drawn to scale for visualisation purposes.

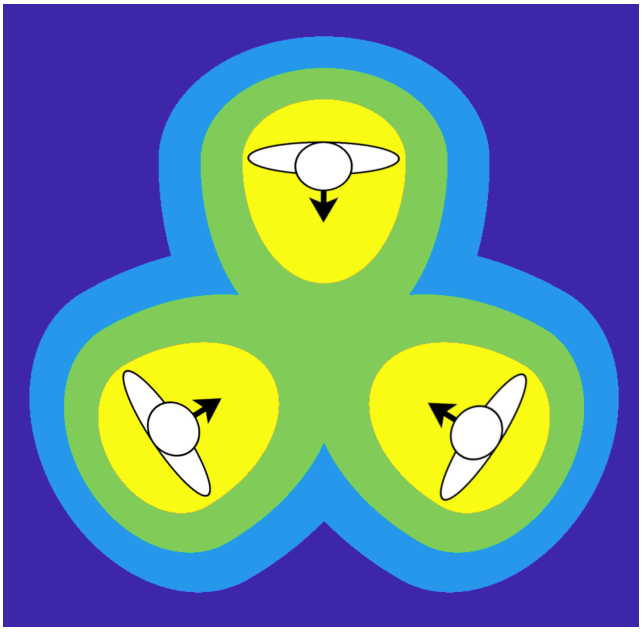


Figure 2.2: Using the egg shaped proxemics to show the social interactions. Proxemic zones are not drawn to scale for visualisation purposes.

The proxemics described by Hall [7] are concentric circles which may differ depending on the situation, as described by Hayduk [8]. A version of this is shown in figure 2.1. Here it is described how people require more space in front of themselves, meaning that invasion of space from the front are viewed more negatively. This results in a egg-shaped social zone similar to the one shown in figure 2.1b. Gérin-Lajoie et al. [9] further argues that the social zones are asymmetrical depending on the dominant side of the body as shown in figure 2.1c.

Barnaud et al. [10] try to determine parameters for the egg-shaped model by using experimental data from Efran et al. [11] to which they fit Rapidly-exploring Random Tree (RRT) based simulated paths using different parameters for the personal and different parameters and models for the interaction space. Comparing their experiments with and without interaction space they conclude that omitting the interaction space is only marginally decreasing the model fit and that therefore the detection of the interaction between two humans might not be as important in their specific application of passing a two person interaction in a corridor. Furthermore they confirm that the personal space should be egg shaped and elongated in the direction the person faces to achieve human-like motion. Figure 2.2 shows the social interaction using only egg shaped zones.

When people interact with other people they tend to establish new proxemics zones that include space between them. Kendon [12] describes different spaces

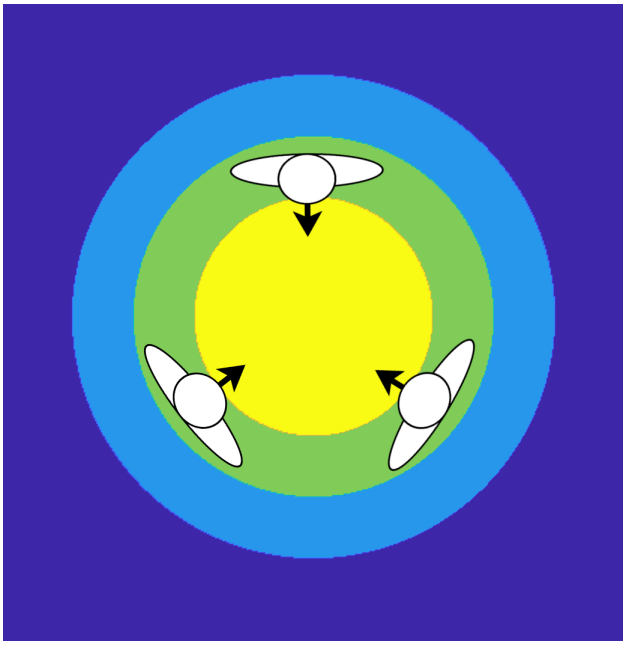


Figure 2.3: Social interaction F-formation. O space is in yellow between the participants, P space is in green and is where the humans are located and r is around the humans in blue, where potential participants can enter. The spaces are not drawn to scale but serves visualisation purposes.

that are relevant when observing different social interactions. Here the O space and P space are described as the space for the activity and the space surrounding the activity and the people, respectively. Additionally, the interaction will have an r space around it, where the members of the interaction will be aware of others who might join the social interaction. This setup is also known as the F-formation, which can be seen in figure 2.3.

Greenberg et al. [13] describes how five dimensions can be used to describe the interaction between human and other objects. These dimensions are Distance, Identity, Location, Movement and Orientation. The distance describes the physical distance between human and object and is related to the zones of proxemics. The identity describes the individual person and can be used to describe spatial interactions between a person and an object. The location is used to determine the type of interactions that are present in a given context. The movement is used to describe the physical movement of human and object over time. The orientation is used to describe what object the user is facing which in turn is used to determine interactions.

2.2 Semantic mapping

A method for understanding the context of human activity is to understand the human semantics of a given area. This can be done with semantic mapping where a given area is assigned a category. This category can then be used to assess the context types that are expected or possible. Knowing the semantic of a given area can help in achieving a more suitable navigation of a mobile robot by assuming human interactions more generally in certain areas [14].

The methods for developing semantic maps varies with different use cases and it is therefore relevant to understand different methods to choose a method of approach when developing a new system using semantic mapping. For tasks where the robot collaborates with a human it is important that the robot understands the semantics of the humans when commanded to certain areas. This is especially true when relying on Natural Language Processing (NLP) where the robot must be able to perform a command issued by the human using verbal communication [14]. When using semantic mapping in outdoor settings the semantics of a given area can be used for outdoor localisation, assessment of traversability, behaviour selection, etc. [15].

According to Kostavelis et al. [14], a problem within semantic mapping is that of evaluating a given method due to a lack of a comparable ground truth. Therefore, there is a need for commonly used datasets to compare different methods reliably.

Garg et al. [16] proposes a taxonomy for robotic semantics that divides into four categories that either extract semantics from a scene, uses semantics to achieve a goal, or does a combination of both. These categories are as follows:

1. Static and un-embodied scene understanding: Using methods to extract information of a given scene from a static camera, separating it from a robotic agent that is learning or using the environment. This is relevant due to the use cases within robotics even if the research is proposed for other applications.

2. Dynamic Environment Understanding and Mapping: Using different types of sensors on a mobile robot enables the possibility to determine the semantics of a robots environment to determine the map using Simultaneous Localization And Mapping (SLAM) techniques.

3. Interacting with Humans and the World: This area connects the ability to perceive the environment with the ability to act within that environment. This takes into consideration the difference between perceiving interaction and perceiving in order to interact.

4. Improving Task Capability: Using semantics in order to improve the capability of other tasks. This can be using semantics in order to improve localisation of a robot in an outdoor setting.

2.2.1 Static and un-embodied scene understanding

Within semantic mapping an important aspect is that of recognising the environment as a given category. To this extent different methods have been utilised to achieve this goal. Du et al. [17] presents a framework for utilising multiple modalities to determine the semantics of a given area by combining Red-Green-Blue (RGB) images and depth data.

Chang et al. [18] describes method for understanding the event present in a given series of images or video to classify the context of the event. This can be an event such as a birthday party or marriage proposal. Such systems can help in natural language understanding of semantics in robot situations, as described by Garg et al. [16].

An important part of scene understanding can be that of scene representation. Here the application can vary and therefore the type of representation can be tailored to a given scenario where pixel wise segmentation or region level semantics can be useful for representation.

Another method of scene representation is scene graphs which describes the relationship between objects in a scene. This can be used to determine how a scene can be divided into areas of related objects [19, 20, 21].

2.2.2 Dynamic Environment Understanding and Mapping

In mobile robotics, SLAM has long been a subject of research due to the difficulty of achieving perfect localisation in highly dynamic environments. Therefore, using semantics to further understand a map for localisation can enable robots to localise themselves within such a map [22, 23].

A hybrid map between geometric maps, generated by distance sensors, and topological maps, developed with certain areas as nodes with edges between them, can be used to achieve semantic localisation. Yue et al. [24] describes a method for using multiple mobile robots to generate local semantics maps that can be combined to a global semantic map.

Different methods have been developed to determine the classification of an area, such as using a Convolutional Neural Network (CNN) to determine the ontology of a given area from the occupancy grid [25] or by using the objects in a scene for classifying the environment [26].

An important aspect of performing semantic mapping is that of ensuring that unseen areas and areas dissimilar to those of training data can also be classified, as described by [27, 28, 29].

A scene graph for interaction of an agent is of use when planning movement, as described by [30, 31]. These scene graphs can then be used to determine relations between objects in a given area and determine the context for robotic use. Especially spatial relations and affordance spaces are of interest when calculating movement with or around such objects. Rosinol et al. [32] proposes Dynamic Scene Graphs (DSG) which can determine spacio-temporal relations between objects within different areas to support moving agents, such as robots, in planning actions and movements.

2.2.3 Interacting with Humans and the World

As described by Garg et al. [16], a robot must understand the semantics of the world and the actions of the humans in that world in order to achieve a successful mixing with humans. In this context successful refers to an interaction that is perceived positively by the humans involved.

Carreira et al. [33] show the importance of gathering data in order to train a robust network for spatio-temporal recognition of human actions. The Kinetics Human Action Video dataset has since been updated multiple times to enable even stronger training of spatio-temporal action recognition [34, 35, 36].

Fang et al. [37] uses reasoning about different objects for grasping to understand the possible interactions with object. This method can be used in collaboration with semantics to understand the context of a given situation for task completion.

Dogar et al. [38] introduce goal directed behaviours, where the robot needs to interact with its environment in order to achieve said goal. These behaviours are based on linked affordances, as defined by J. J. Gibson [39].

Tellex et al. [40] introduces the concept of inverse semantics where the robot recognises its inability to perform a given task and asks a human for help. This enables the robot to resume autonomy when assisted by the human. Here NLP is used to request help. This idea is extended by Gong et al. [41] to include temporal and spatial aspects in the requests. This means that the robot is able to ask about objects in certain locations or describe where a given object used to be located.

2.2.4 Improving Task Capability

For robot navigation, the use of semantics to improve the performance of different types of tasks can be of great value. Therefore, understanding how to use semantics is just as important as developing the systems for perceiving semantics.

One use of of semantics is within visual place recognition, where the type of environment is used for localisation of the robot [42]. This can be used to determine if the robot has seen the location before.

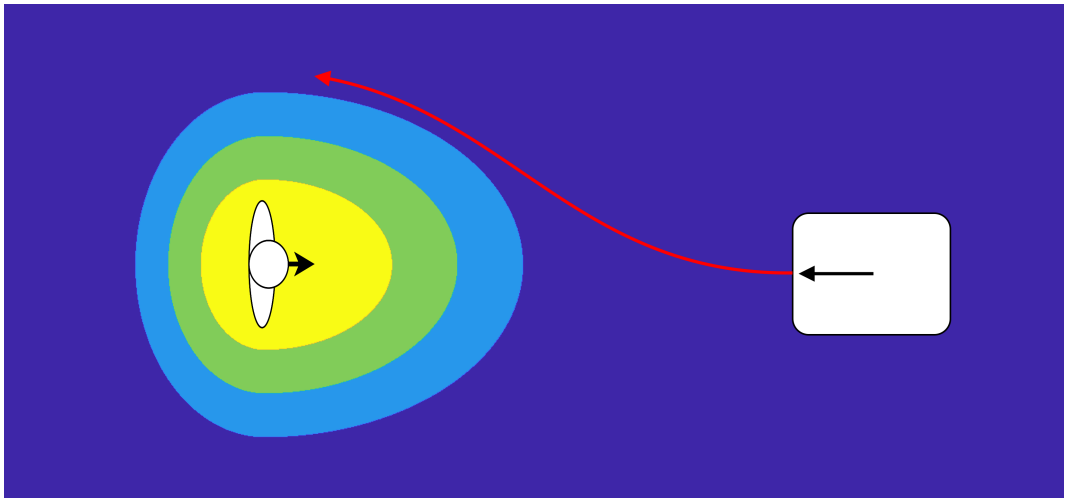


Figure 2.4: An example of a mobile robot planning its path along the social zones of the human to achieve a comfortable interaction. Proxemic zones are not drawn to scale for visualisation purposes.

2.3 Human Robot Interaction

When humans and robot work side by side, whether it being in industry, service, or out in the open, it is important that the interaction is perceived positively by the human(s) interacting with the robot. Therefore, it is essential to use metrics to determine how the robot is perceived, as well as developing tools for improving any issues that may cause a bad interaction. Savela et al. [43] describes how acceptance of robots in social interactions is essential, especially when the robot is used as assistance or as a co-worker. Fuji et al. [44] highlights the importance of knowing the capabilities of a given robot for acceptance. Here the users describe how they are unsure of using the robot due to unknown actions. It is therefore important to ensure sufficient training of staff or developing a robot that exhibits its capabilities. Law et al. [45] show how important the movement of a robot, whether humanoid, animal-shaped, or any other shape, influences the trust of said robot. It is found that movement that follows expected patterns and social standards improves the trust and comfort around the robot which is important to ensure that humans want to use the robot. An example of following social zones can be found in figure 2.4.

Mavrogiannis et al. [46] describes the importance of understanding humans as more than just dynamic obstacles in an environment to achieve more than just collision avoidance. Assuming that the human will seek to avoid collision will improve the trajectory of the robot, resulting in less oscillatory movement.

Babel et al. [47] takes another step in understanding human behaviour by analysing potential conflicts between humans and robots when working in the

same environment. The authors argue that a robot that always yields can become inefficient in the long run and therefore must be able to interrupt the user. Different methods have been tested for an assertive robot which shows that being polite and using knowledge from psychology results in higher trust from the user [48]. Furthermore, it is found that adapting to the compliance of the user is important in order for a successful collaboration between human and robot. Nomura et al. [49] found that using culture appropriate methods for asking resulted in faster and more accepted interactions. Therefore, it is important to ensure a polite robot when being assertive of a certain task in order for a successful and positive interaction [50]. It can, however, be important to determine the sufficient amount of politeness combined with requests in order to achieve an effective interaction [47].

Turnwald et al. [51] expand the collision avoidance problem during the motion of a mobile robotic platform using human like motion planning based on game theoretic decision making. Testing their planner using a Turing test in Virtual Reality (VR) participants where not able to distinguish between the planned motion and real human motion, whereas the participants could distinguish between human motion and established concepts of obstacle avoidance based on e.g. social forces.

2.4 Scene understanding

To navigate in an environment filled with different dynamics objects and humans moving around performing different tasks, scene understanding is necessary. This means that a mobile agent must be able to identify objects in the environment and determine the physical location in relation to each other. This is essential for both object avoidance as well as finer interaction with humans and objects. Dovesi et al. [52] propose a method for performing stereo matching and semantic segmentation in a combined manner. Using this method they achieve semantic stereo matching in real time on a Graphics Processing Unit (GPU) or with lower accuracy using embedded devices, which enables the use in robotics applications.

Scene graphs can be used to describe relations between objects in an environment. Rosinol et al. [32] describes dynamic scene graphs as a way to describe the relationship between dynamic objects as they interact in a given scene. This includes potential actions which can be used for planning movement of a mobile agent. As described by Chang et al. [21], the challenge in scene graphs is the potential size of the graph. This is due to the fact that objects and humans can have many relationships and therefore many edges will be drawn in the graph. This can be especially challenging when working with objects that are far from one another or unknown objects.

Wong et al. [53] describes the importance of understanding unknown objects in a scene in order for the robot to behave in an expected manner. This means

that dynamic objects that are not identified should still be treated in an appropriate way in a given environment. To achieve this an object is segmented into an unknown class in order for the robot to enable interaction - a challenging but important task.

2.4.1 Perception of human activity and groups

An important aspect of scene understanding in robotics is that of human activity recognition. When a mobile robot traverses an environment with humans it is therefore important to understand the activity that the humans to pass are performing.

Kostevalis et al. [54] used a skeleton-based approach to understand the activity of the human by combining the actions of the skeleton with the associated object in a spatio-temporal network. A behaviour understanding pipeline is introduced, based on a Dynamic Bayesian Network (DBN) that is based on Interaction Units (IU)s (introduced by Ryu et al. [55]) that describe the transition of a state from a defined start to a goal that are connected through a so called mental process. In their implementation the state is the environment and the mental process is an action of the user. Therefore they describe the behaviour by a series of IUs and can therefore classify this series using the previously mentioned DBN.

Taylor et al. [56] describe the need for new algorithms to achieve activity recognition from a robot-centric perspective. This means that only sensors mounted on a mobile robot should be used to determine the activity of the humans surrounding the robot without using outside information. This should enable the robot to determine the actions of the humans in order to interact and collaborate. However, Taylor et al. [56] describe how current research is conducted in well-controlled environments which means that only very specialised tasks can be detected and interacted with by the robot. This means that the models may not generalise well to open areas where humans may perform unpredictably compared to the controlled environment. Therefore, it is important to develop a method that can generalise to such situations without requiring large amount of annotated data. Sadeghian et al. [57] uses social and physical cues to understand the potential paths of humans walking in a scene. Vemula et al. [58] developed Social attention to predict the trajectories of humans walking among each other.

Li et al. [59] describes the necessity to understand the social relationships between humans in a scene to understand the interaction. This means that understanding the relationship will enable a system, such as a robot, to predict the actions of interacting humans. Zhang et al. [60] high-light how CNN for object detection and pose estimation to find the relationship between humans and objects. Ji et al. [61] describes a framework for understanding actions in a video by by using a scene graph of objects.

Song et al. [62] describes the use of skeletons from key point estimation to understanding the action of a human even when occlusion occurs in the stream of images. This is important when movement occurs because of the inevitable occlusion found in dynamic environments.

2.5 Scope

From the analysis performed in this chapter, it is evident that there are many approaches to context aware navigation. Therefore, it is important to define the scope for developing such an approach.

As described by Youssef et al. [63], no strict definition of social robotics are made. Some papers do not specify their definition while others use different definitions for various purposes. The properties usually associated with social robotics are: autonomy, social awareness and reaction to social situations, communication, sensing of human gestures and reaction to human activity. These properties point the necessity of having social robots that can perform their service while adhering to social norms.

In this project the limit will be on developing a pipeline that enables the use of social robots in an industrial context where humans may be interacting with each other in different locations and different group sizes. Here the robot must be able to identify when a social interaction is happening between humans and plan its route around that interaction even when the people are standing at further social distance (as described in section 2.1.1). Furthermore, the robot should enable planning around the social zones of the human where the plan is adhere to the social zones as the environment allows it.

To ensure that the developed system will be relevant for newer generations of robotics, it should be implemented with the current methods used for robot communication. Therefore, the system should be implemented in a modular fashion where different modules can be exchanged with others without changing the modules. Therefore, this project seeks to implement the system as nodes using Robot Operating System 2 (ROS2) to ensure that the developed modules can be used in the foreseeable future. This means that the developed nodes should be integratable into a navigation stack without limiting the use of path planners or controllers.

As the detection of people in the environment is crucial to achieve context aware navigation in a people filled environment, the detection of the people must be able to detect humans in the surrounding area of the robot. Therefore, this project will focus on developing a method for detecting humans with sensors mounted on the mobile robot. However, due to the modular setup of the system, the nodes responsible for detecting interactions between people should be agnostic to the method of detecting people as this will depend of the robot setup. Therefore,

the people detection module should be able to detect the humans in a global map where the location and orientation of the human can be represented.

As the types of interactions can be very diverse it is important to limit the types to those relevant for the project. Therefore, this project only considers Human-Human Interaction (HHI) where between two and four people are standing across from each other or in an F-formation, as described in section 2.1.1. This means that the system must be able to detect the humans and ensure that the defined social zones are adhered to, when no interaction is present and otherwise consider the interaction zones in the navigation.

The robot will be limited from interacting directly with the present humans by communicating but will rather change its path as needed to avoid disrupting interactions.

Furthermore, the robot should be able to handle changes in interactions and must therefore be able to update the available area for navigation while traversing. Therefore, the developed modules should be able to run at a similar rate to the rest of the navigation control loop to ensure updated paths when necessary.

2.5.1 Requirements

The considerations above lead to the requirements of this project which should be evaluated to determine the success of the implemented system. The requirements are as follows:

a The system must be implemented using ROS2 and should be compatible with the navigation stack by receiving velocity commands and supply sensor readings.

b The nodes should run at a rate of at least 10hz to keep up with the calculation time of the navigation stack. This means that the accumulated runtime of the nodes should be less than 100ms

c The system must be able to detect the location and orientation of humans in front of the robot between 1m to 5m to ensure safe navigation.

d The system must be able to develop proxemic zones for the detected people to ensure navigation outside of these. Here the numbers described in section 2.1.1 are used.

e The system must be able to detect HHI and alternate its path to traverse outside of the interaction zone.

The requirements lead to the final problem formulation which guides the rest of the project. From this a technical analysis will be described in the following chapter, which is used to highlight the used technologies to perform context aware

navigation in an industrial setting. The highlighted technologies can be exchanged for similar systems to change the performance of the robot in the future, if needed. The final problem formulation reads as follows:

How can a modular system for context aware navigation be developed for edge computation on a mobile robot?

3 - Technical Analysis

The technical analysis seeks to cover the theoretical methods used in this project. This means that this chapter describes the background of the methods that are used or implemented in chapter 4.

3.1 ROS

Robot Operating System (ROS) is a middleware suite for software development on robots [64]. It offers infrastructure and a large collection of open source developed libraries for the use with all sorts of robots, be it for example manipulators or mobile robots. The latest release version of ROS is "ROS2 Humble Hawksbill" [65]. Here, ROS2 stands for the second generation of ROS.

Robot Operating System 1 (ROS1) has reached its final release with the version "ROS Noetic Ninjemys" and will only be supported until may 2025 [66]. As mentioned, the official development is being continued in the second generation of ROS, ROS2.

The computation architecture in ROS2 is build on so called "lifecycle nodes" [67]. These nodes contain the user application that are based on the programming language specific API (python, C++ etc.), that are all linking to the same underlying C library. These nodes can then be grouped in the form of a package in order to install, use and share them [65].

The nodes are communicating using an asynchronous publisher/subscriber structure using a DDS based middleware. Here, nodes can publish messages of specific types on topics. These topics are then used to identify the message from another node.

In addition to messages, ROS2 includes Services and Actions as a form of communication between nodes. Services are implement as a request-response type communication, such that nodes can actively request something from other nodes without needing to send a message over the topic based asynchronous communication. This also simplifies to associate a request and response.

Actions however, are another asynchronous communication method, whereas they extend the request and response style of the services to request and response and periodic feedback. Therefore actions are better suited for long enduring tasks where one node needs to observe the progress of another.

Another structure offered by ROS is the library "tf2". tf2 is the second version

of the geometric transformation library "tf" [68] and allows to build a tree like structure of coordinate systems that are connected via geometric transformations, called tf_frames and tf_transforms. As a consequence this allows to transform sensor data from one tf_frame to another and with the implemented buffer of tf_transformations also in time.

3.1.1 Nav2

ROS builds strongly on community based development that means however, that a majority of the community packages is only available for older releases, as the developers are not forced to maintain and port their packages across releases. However the development of official packages like for example Navigation 2 (Nav2), the successor of the ROS1 navigation stack [69], is only taking part in the latest release versions. Nav2 offers a framework for grid map based navigation and separates it in to a planning and controlling stage. In contrast to the ROS1 navigation stack [70], Nav2 relies on behaviour trees to manage situational navigation using for example recovery behaviours. This allows the navigation stack to achieve long running task where multiple behaviours can be performed in parallel without blocking each other. Additionally, the behaviour tree can supply feedback to clients of the progress of a given task. This can be done both synchronously and asynchronously. In the behaviour tree multiple action servers handle different parts of the navigation where different modules, such as costmaps, can be added or exchanged with others. This also allows for tuning of different parts of the navigation stack which allows for modification in robot movement [71].

Furthermore, Nav2 supplies a great collection of continuously maintained planners and controllers, whereas the local and global planners in the ROS1 navigation stack are partly unmainted and or deprecated.

In order to develop software for robotic applications, analysis tools are of great value. Here ROS offers a great selection of visualisation tools like RViz (in ROS2 RViz2), plotjuggler and rqt based dashboards and control panels [72, 73, 74].

3.2 CoHAN Planner

Singamaneni et al. [75] developed the CoHAN Planner for Human-Aware navigation using ROS. The planner was designed to handle several different interactions between humans and robots in both indoor and outdoor settings considering human motion and social aspects. The system has multiple modes of planning to address the varying interactions between humans and robots. The authors showed that the system performs well in different scenarios compared to similar local planners.

CoHAN uses an architecture based on the ROS navigation stack. Here the

stack is modified to include a *Human Safety Layer* and a *Human Visibility Layer* in both the global and local costmap. Furthermore, a Human Path Prediction module is implemented to predict the paths of humans for planning of robot motion. The planner uses the HATEB local planner [76]. HATEB uses the state of the human and the planner to determine the planning mode of CoHAN and transition between planning modes. The modes of planning are as follows:

1. **SingleBand mode**: The initial state of the planner with just an elastic band mode for the robot. The planner stays in this mode when no humans are within a 10m radius of the robot.

2. **DualBand mode**: This mode is used to handle dynamic humans. This mode enables the robot to handle changes in human path and considers the closest two humans while planning.

3. **VelObs mode**: This mode uses the human aware criteria but only adds elastic bands to humans if they have velocity.

4. **Backoff-recovery mode**: The final mode is used when there is no valid solution to the planner unless an agent, either human or robot, clears the path. This situation happens in corridors, where the robot can not traverse next to the person safely. The system gives priority to humans and backs off until the path is clear for the human.

To ensure navigation that is comfortable for the humans close to the robot, the planner uses two additional social constraints, *Visibility* and *Relative Velocity*.

Visibility adds cost to the optimisation of the plan when the robot moves behind the person and suddenly becomes visible. This is done to ensure that the robot does not startle the human. The constraint is implemented by using a 2D half Gaussian behind the person.

The Relative Velocity constraint adds cost to the plan depending on the relative velocity between human and robot in addition to their distance. This causes the robot to have low velocity when closer to humans if a path with larger distance is not possible. This helps force the robot to choose plans with larger distance.

3.3 PersonLab

Papandreou et al. [77] presented PersonLab, an approach for box-free bottom-up pose estimation of multiple people in images. PersonLab is an efficient single-shot model that handles both semantic-level reasoning and object-part associations. The model uses a CNN to detect keypoints and predict displacement of these, which allows for grouping keypoints into a person pose.

Contrary to a top-down approach, where a person is first localised using a bounding box detector, a bottom-up starts by localising individual keypoints before grouping these into people instances.

PersonLab detects keypoints by using a encoder-decoder architecture to generate heat maps determining where keypoints are located in a given image. Using local maxima of the heat-map allows for detection of different keypoints that can be used for grouping of keypoints into poses. A pairwise connection of keypoints is used to connect keypoints into poses, by pairing e.g. *right elbow* with *right shoulder*. To ensure that the same keypoint of a person is not detected multiple times, keypoints of the same type are compared to each other and a non-maximum suppression is performed for keypoints within a given radius.

Due to the bottom-up approach of PersonLab, the algorithm can handle occlusion reliably by simply connecting visible keypoints only. This makes the algorithm robust in visually noisy environments.

3.4 Tracking

As described by Li et al. [78], a reliable tracker is necessary to achieve information on multiple tracked targets. Using methods such as a Kalman Filter (KF) is useful for keeping track of a single object [79]. However, when tracking multiple objects, it is important to ensure the right association between tracklets. Therefore, using a method like the Munkres algorithm to assign detections to new tracks can be useful.

The Munkres Algorithm and the KF are explained in the following sections.

3.4.1 Munkres' algorithm

Munkres' algorithm [80] is used for the assignment problem of minimising cost. In this setup only one object can be assigned to one previous track which means that each detection should be assigned just once. Performing this assignment in a brute force manner requires $n!$ operations making the scalability perform slow for multiple objects [81]. Munkres' algorithm performs the assignment in $O(n^3)$ time by performing 6 steps on a $m \times n$ matrix with n tracklets and m detections. As the algorithm was developed in 1957, it is assumed that the assignment is done by hand, hence the language of the steps refer to this [81]:

1. For every row in the matrix, subtract the smallest element of that row.
2. Choose a zero (Z) in the resulting matrix. If there is no starred zeroes in the row and column of Z, star it. Repeat for every element in the matrix.
3. Every column containing a starred zero is covered. If all columns are covered the algorithm is done. Otherwise, proceed to step 4.

4. Choose an uncovered zero and prime it. If no starred zero is found in the row of the primed zero, go to the next step. Otherwise, cover the row and uncover the column of the starred zero. Continue until all zeroes are covered. Save the lowest uncovered value and go to step 6.

5. Create a series of zeroes alternating between primed and starred in the following manner: Z_0 is the uncovered zero that was primed in step 4. Z_1 is the starred zero of the column of Z_0, if one is available. Z_2 is the primed zero in the row of Z_1. This continues until a primed zero has no starred zero in its column. Then all starred zeroes are unstarred, each primed zero is starred, all primes are erased, and every line is uncovered of the matrix. Return to step 3.

6. The value found in step 4 is added to each element of the covered rows and subtracted to each element of uncovered columns. Return to step 4.

When done, the starred zeroes indicates the location of the assignments in the original matrix. With assignment between tracked objects and new detections performed, the tracker can update the tracklets with for example Kalman filters, as described in the following section.

3.4.2 Kalman filter

The Kalman filter keeps track of different states of an object. This is done by utilising a recurring method for estimating the dynamic state, especially when noise is present in the measurements. By maintaining estimates of the state vector ($\hat{x}$) and the error covariance matrix (P) of a given system, the Kalman filter updates the system over time with measurements and predictions. This is done by assuming a normal distribution of the potential state of the system.

The filter uses two stages to estimate the state of the system, prediction and update. Prediction takes the current state of the system and estimates the state at the new timestep. This is done with $\hat{x}(k+1|k)$ and $P(k+1|k)$. In the update step the actual measurements of the system are compared with the estimate from the prediction step to calculate the new position by determining the assumed noise of the measurements. The Kalman filter then weights the prediction and measurements using the calculated Kalman gain (K) to give the new state of the system for the timestep [82].

Prediction As described above, the first stage of the filter, prediction, calculates the estimated state of the system at the following timestep using the following equations:

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) + u(k) \tag{3.1}$$

$$P(k+1|k) = F(k)P(k|k)F(k)^T + Q(k) \tag{3.2}$$

Where:

- $\hat{x}$: State vector
- k: Kalman gain
- F: The state transition matrix
- u: Control vector adding any known changes to the system
- P: Covariance estimate matrix
- Q: Overall process noise in a covariance matrix.

Update The second stage of the filter, update, uses the measurements for estimating the state of the system by combining it with the prediction using the following equations:

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + Ky \tag{3.3}$$

$$P(k+1|k+1) = (I - (K \cdot H(k+1))P(k+1|k) \tag{3.4}$$

Where:

- y: Measurement residual
- I: Identity matrix
- H: Vector mapping matrix

The Kalman gain, K, is calculated by $K = PH(k+1)^T S^{-1}$, where $S = HPH^T + R$ is a is covariance matrix that increases with increasing measurement noise in matrix R. Furthermore, y is calculated as $y = Z(k+1)^T - H(k+1)x$, where Z is the measurement of the state.

3.5 YOLOv7

As described in section 2.4, it is important to enable a context aware system to detect different objects in a scene to determine interactions. YOLOv7 [83] enables detection of any arbitrary object that it has been trained for. YOLOv7 builds upon the advances found in YOLOv4 [84] to achieve fast and accurate object detection in images. The model enables real-time detection on edge devices and uses different

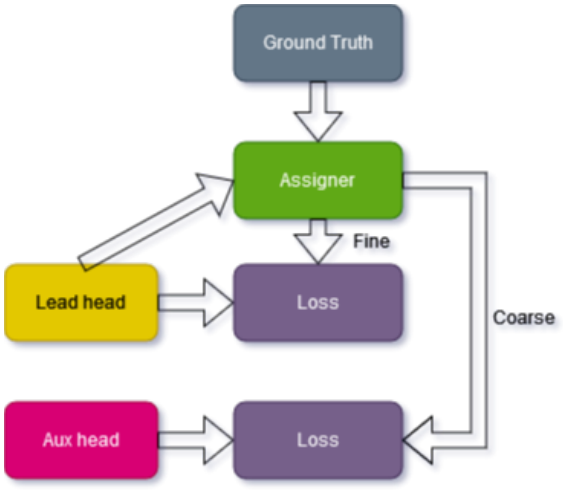


Figure 3.1: The use of a soft label assigner to train both the lead and auxiliary head. The label assigner can be changed to allow more relaxed constraints for the coarse label while providing a harder label for the fine head. Inspired by [83].

methods for improving accuracy without sacrificing inference cost. These methods are developed in YOLOv4 and improved for YOLOv7 and are known as Trainable Bag of Freebies (TBoF). These include re-parametrisation where multiple trained models are used to calculate the final set of weights for the model. This increases training time but results in the same inference time, as the amount of trained weights remains the same.

The YOLOv7 model further developed the Extended Efficient Layer Aggregation Network (E-ELAN), a new computational block that enables the network to learn features more reliably by keeping the original gradient path while changing the computational block. This enables the network to stack more blocks by expanding the cardinality of the computational block which increases the number of parallel convolutions in the block. This means that the block will perform computations more efficiently by aggregating feature maps to enhance quality while keeping computation at the same level.

Furthermore, YOLOv7 has multiple heads for output prediction by enabling multiple predictions during training. This improves training by having course and fine predictions. The auxiliary head is found earlier in the network and uses features found at this level to guide the lead head. The training process utilises information obtained by the lead head to update the auxiliary head. This allows the lead head to concentrate on residual information that has yet to be learned. Figure 3.1 shows the label assigner that generated soft labels for training both the lead head and the auxiliary head.

YOLOv7 manages to achieve a 75% reduction in parameters and 36% reduction in computation time compared to YOLOv4. For the tiny version of YOLOv7

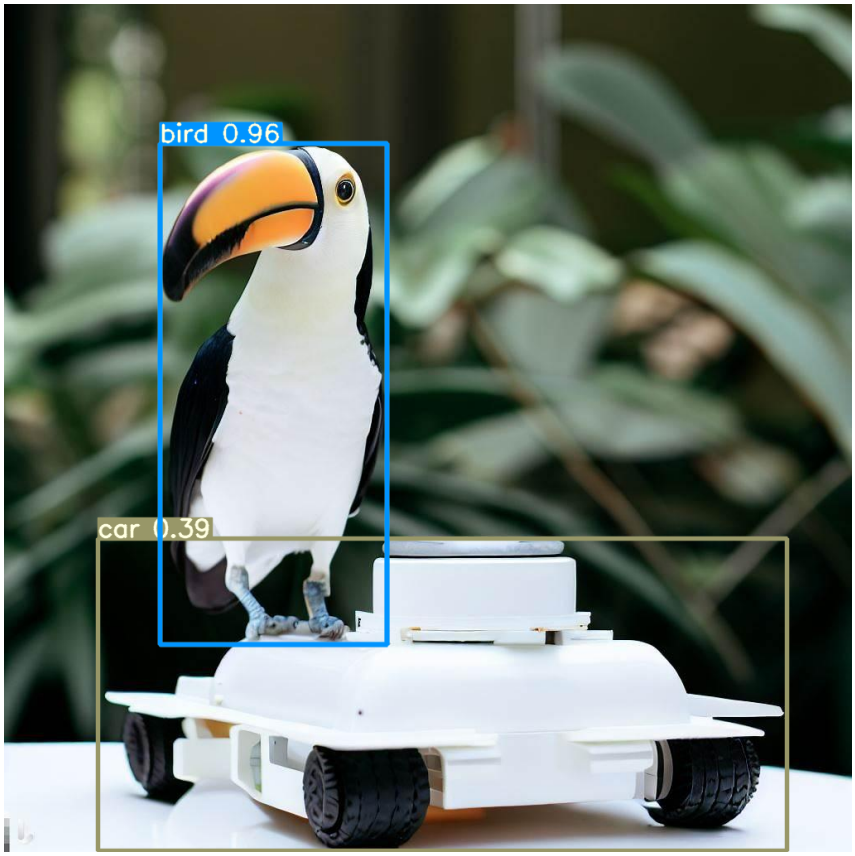


Figure 3.2: An example of YOLOv7 used on the Nav2 Context Aware Navigation (Nav2CAN) image to detect the types on object present. As mobile robots are not part of the MS COCO dataset [85], which YOLOv7 is trained on it is classified as a car with low confidence.

the reduction is 39% in parameters and 49% in computation time while achieve the same Average Precision (AP), compared to YOLOv4-tiny. This means that the YOLOv7 tiny version is even better suited for using on edge devices as the the needed computation and memory is reduced. Using an optimised architecture for edge computing can enable real-time computation which is necessary for achieving a safe navigation with a mobile robot in a human filled environment.

An example of YOLOv7 used on the frontpage image of this project can be seen in figure 3.2.

4 - Design and Implementation

This chapter describes the different modules designed and implemented for this project. The system overview of this project, from now on referred to as Nav2CAN, is shown in figure 4.1. Nav2CAN is separated into three modules the *People Detection Module* (green), the *Context Module* (red) and the *Nav2* navigation stack (blue).

The setup of these modules and their communication with each other is described in the following sections. Furthermore the hardware used to run Nav2CAN on the edge is introduced.

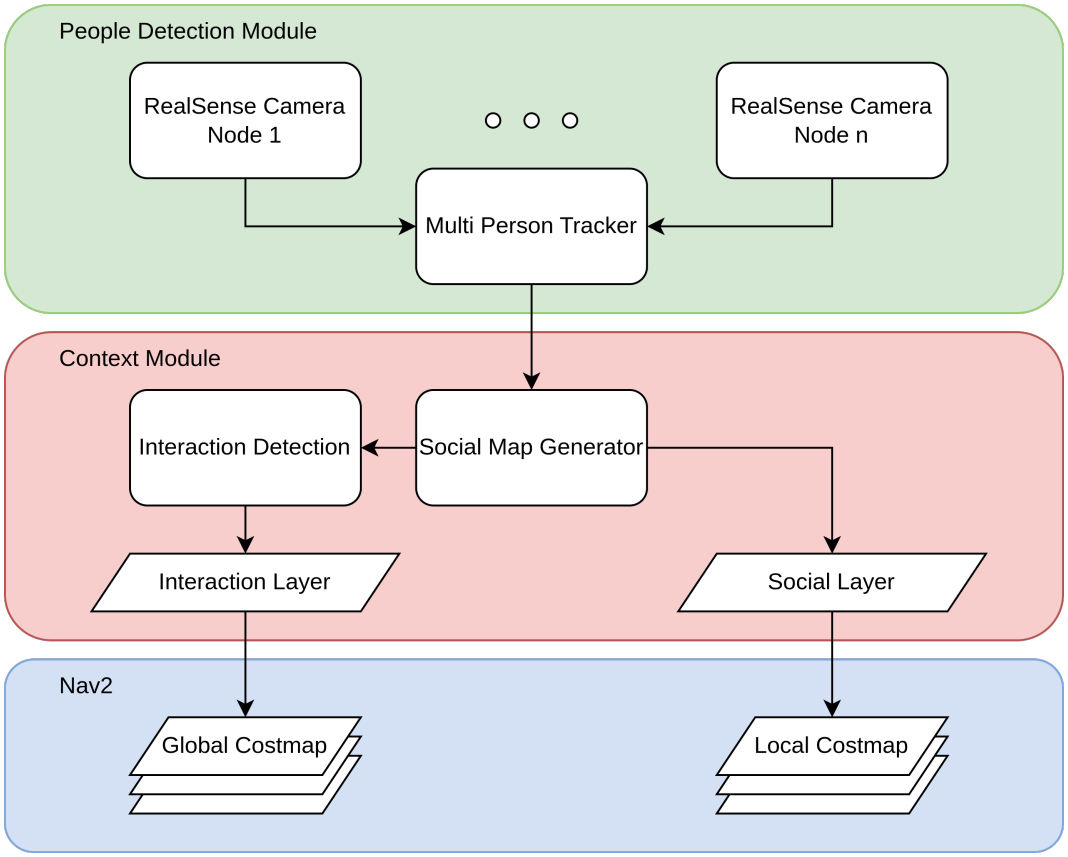


Figure 4.1: Nav2CAN system overview

4.1 Hardware

The hardware of the system is used to run different nodes of the setup. Here the description is made in terms of the needed capabilities which enables alterations or using a different type of unit if needed.

4.1.1 Jetson Orin

The Nvidia Jetson AGX Orin developer kit [86], is used in this project to perform detection and tracking of people, as described in section 4.2. The series of Jetson boards are developed to enable parallel computing on edge devices, such as on a mobile robot. This means that deep learning techniques, such as CNN, can be run at a relatively high rate while keeping power consumption low, compared to using a desktop PC or similar. Furthermore, using the Jetson boards enables the use of the Jetpack SDK for acceleration of software as well as using the jetson-inference package for running optimised neural networks [87]. In order to communicate with the nodes running on the remaining hardware, the Orin is running a docker container with ROS2 Humble.

4.1.2 Robot controller

The navigation stack runs on an extra device. Here a laptop with an *Intel i7 7700HQ* running Ubuntu 22.04 and ROS2 Humble is used, as the integration of the navigation stack on the Orin with all required dependencies is too time intensive for the time spent on Nav2CAN.

4.1.3 Intel RealSense D435

The Intel RealSense depth cameras are developed to perform reliable depth measurements using Infrared (IR) projection and IR imagers to enable the creation of depth images. The Intel RealSense D435 has an RGB camera in addition to the depth module to enable multi modal handling of data [88]. Furthermore, the RealSense has the possibility of aligning the depth image with the RGB image and applying different filters to the output.

4.1.4 MiR100 platform

For the real world part of this project the Autonomous Mobile Robot (AMR) used is a MiR100 [89]. The platform is equipped with two SICK s300 laser scanners [90] for 360° detection around the robot, which are also used for emergency stopping if the robot gets too close to any objects. Figure 4.2 shows the robot used in this project.

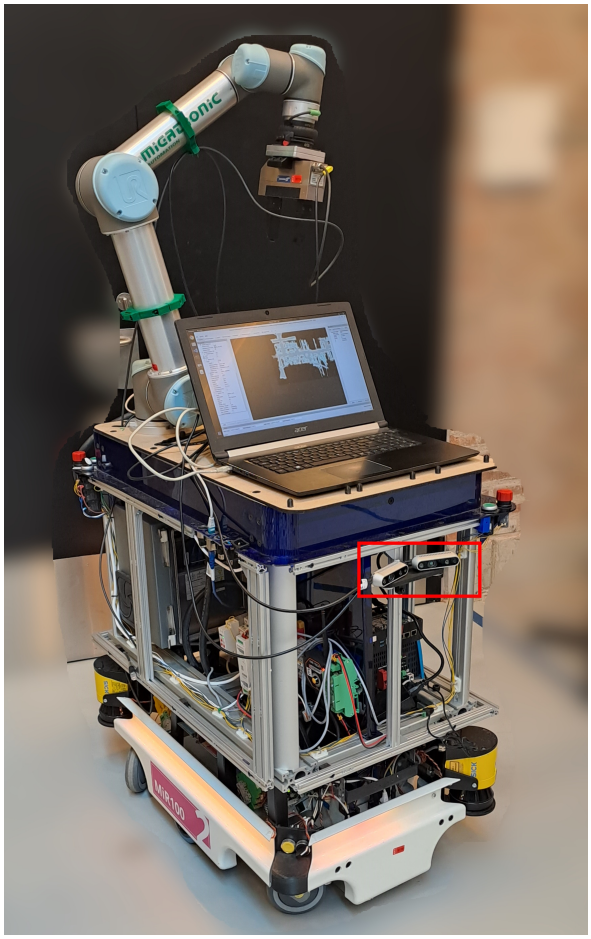


Figure 4.2: The "Little Helper" robot base on a MiR100 platform. The platform is equipped with a UR5 [91] (not used in this project) and modules for using the platform for different experiments. The RealSense cameras are mounted on the front of the robot (shown in the red box). On top of the robot is the laptop used for running the navigation stack for this project.

4.2 People detection module

The people detection module includes the functionality of recording, detecting and tracking people in the environment. It is structured as shown in figure 4.3 and designed to process the data of multiple *RealSense D435* cameras and track the people surrounding them using the *Multi Person Tracker* node.

4.2.1 realsense2_camera node

The *realsense2_camera* node is part of the *realsense-ros2* wrapper which wraps the *RealSense SDK 2.0* and thus offers control of the cameras using ROS2 [92]. By launching the *realsense2_camera* node, the RealSense is automatically detected and

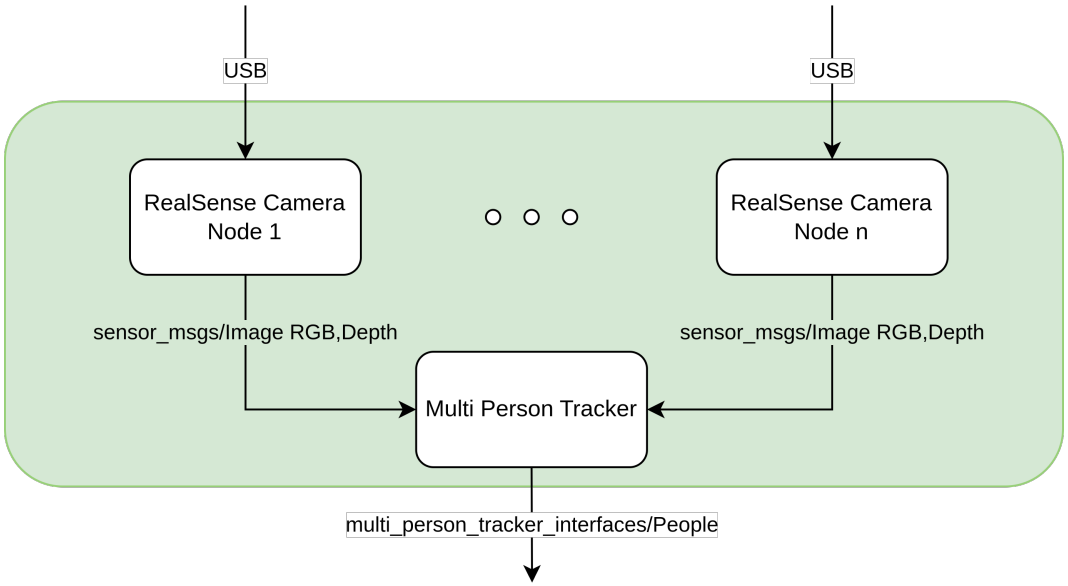


Figure 4.3: Overview of the people detection module

launched with the configuration parameters specified in the launch file. Then the node publishes in the case of this implementation the RGB image and the aligned depth image as a ROS2 message of type *sensor_msgs/Image*.

4.2.2 Camera mounting

In order to use multiple *RealSense D435* cameras, their geometric relations between them need to be known for the entire system.

To define the position in the real world a bracket is developed that fixes the translational relation between two cameras to 110mm as shown in figure 4.5, such that the two cameras don't interfere with each other. However since this only fixes the translational relation the rotational relation has to be determined such that sufficient coverage is achieved.

First the decision is made to have the Field of View (FOV) of the two cameras cross each other in an effort to have more reliable detections in front of them as both cameras will have more overlap in front of them. The overlap of the cameras can be seen in figure 4.4.

The rotational relation between the cameras needs to be calculated. As mentioned in section 2.5.1, it needs to be assured, that a person can be detected, when standing at a minimum 1m away from the cameras. This results in the calculation of the angle between the long side of the *RealSense* camera and the base plate shown in equation 4.1. Where $HFOV_{camera}$ describes the horizontal FOV of one camera, w_{person} is the approximated width of a person, $dist_{cameras}$ is the distance

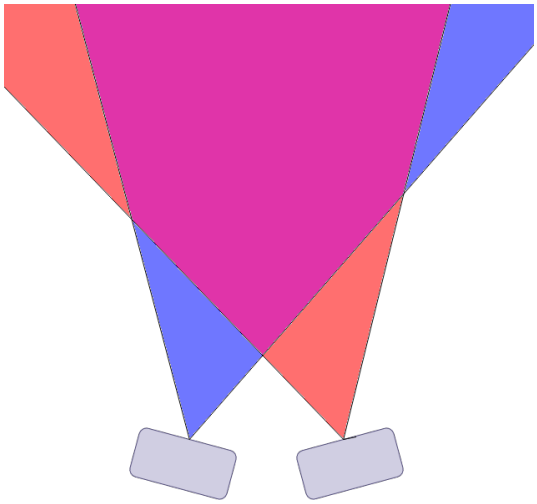


Figure 4.4: The FOV of the overlapping cameras. Blue indicates FOV of one camera while red indicates the other. Purple is the FOV where both cameras overlap.

Figure 4.5: Mounting bracket for two *RealSense* cameras

between the cameras and $dist_{person}$ is the minimum required distance to said person. This also allows to calculate the apprxiamte FOV of the resulting setup using equation 4.2. This however is only an approximation since the cameras are not located at the lower middle intersection point shown in figure 4.4.

$$\begin{aligned} \alpha &= \frac{HFOV_{camera}}{2} - \arctan\left(\frac{\frac{w_{person} - dist_{cameras}}{2}}{dist_{person}}\right) \\ &= \frac{54.732 * \frac{\pi}{180}}{2} - \arctan\left(\frac{\frac{0.5-0.11}{2}}{1}\right) * \frac{180}{\pi} \\ &= 16.33° \end{aligned} \tag{4.1}$$

$$FOV \approx (\alpha + HFOV_{camera}) * 2 \approx 87.4° \tag{4.2}$$

The finished mounting bracket with the two cameras is then mounted on the platform as shown in figure 4.2.

Finally the geometric relations can be implemented in the form of coordinate systems or rather tf_frames such that points can be determined with respect to any other tf_frame of the setup.

4.2.3 Multi Person Tracker

The *Multi Person Tracker* node offers the detection and tracking of people located in the FOV of the RealSense cameras. This is done using the following functionalities.

- Keypoint estimation
- 3D point calculation
- Orientation and Position calculation of person
- Determining the 2D map pose of the person
- Tracking of the person

Each functionality will be described in the following subsections to give an overview of the whole module.

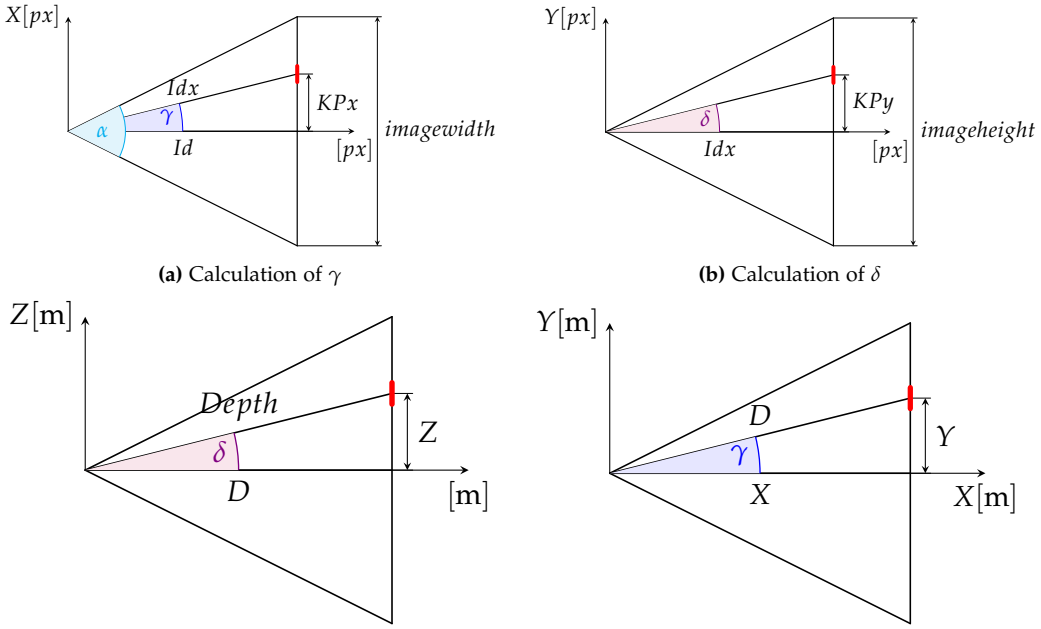
Keypoint estimation

For detecting keypoints of people, this project uses poseNet [93] from the Jetson-inference library [94]. poseNet uses TensorRT [95] to run an optimised network on the GPU found on the Jetson Orin, described in section 4.1.1. poseNet can be used with multiple pre-trained models from Nvidia or use a custom model. poseNet is based on the PersonLab model, described in section 3.3.

3D point calculation

After estimating the position of the keypoints in the rgb image, the 3D cartesian position of said keypoints has to be determined. The RealSense cameras has the option of aligning their depth image to the rgb image before publishing both, as described in section 4.1.3. This results in the ability to get the depth information of a rgb pixel with the same coordinates in the depth image.

Having both, the X and Y position in the image plane and the depth allows to transform the position into 3D cartesian space using trigonometry according to figure 4.6. To reduce the effect of the rather noisy depth image of the RealSense camera, not only one pixel of the depth image is taken into account, but rather the median of the distances in a 3×3 square area around it.



(a) Calculation of γ

(b) Calculation of δ

(c) Vertical projection and calculation of Z

(d) Calculation of X and Y

Figure 4.6: Depth to cartesian coordinates projection. The exemplary position of the keypoint corresponds to a position in the upper left section of the camera image picture in red [3]

As shown in subfigure 4.6a, the distance Id to the imaginary image plane can be determined using half of the *imagewidth* along with half of the field of view α using equation 4.3. Then the angle γ can be determined using equation 4.4 and consequently the hypotenuse Idx can be calculated using the Pythagorean theorem according to equation 4.5.

$$Id = \frac{imagewidth/2}{\tan(\alpha/2)} \tag{4.3}$$

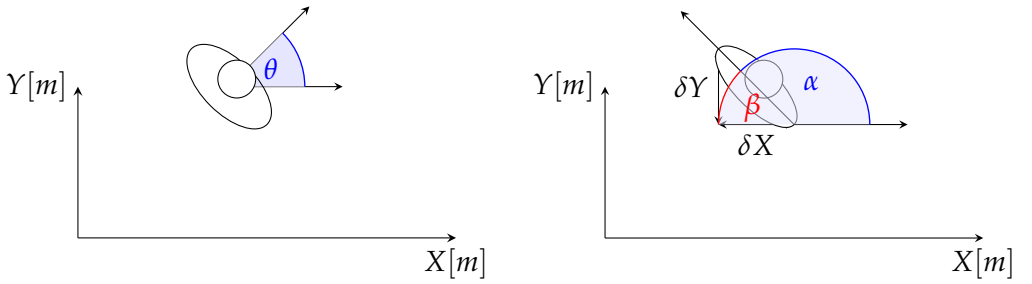
$$\gamma = \arctan(\frac{KPx}{Id}) \tag{4.4}$$

$$Idx = \sqrt{KPx^2 + Id^2} \tag{4.5}$$

Using Idx from equation 4.5 the angle δ can be determined according to subfigure 4.6b and equation 4.6

$$\delta = \arctan(\frac{KPy}{Idx}) \tag{4.6}$$

Having determined both angles δ and γ based on the image plane, these can be transferred to the cartesian space and applied to the median depth measurement *Depth* of the keypoint to determine the cartesian coordinates.



(a) Orientation of a person **(b)** Calculation of orientation using specific keypoints

Figure 4.7: Calculation of the orientation of a person based on keypoints

According to subfigure 4.6c the distance D to the plane through the keypoint that is orthogonal to the view direction of the camera can be determined according to equation 4.7.

In addition the cartesian coordinate Z of the keypoint can be determined according to equation 4.8.

$$D = \cos\delta * Depth \tag{4.7}$$

$$Z = \sin\delta * Depth \tag{4.8}$$

Finally the remaining cartesian coordinates X and Y can be calculated as pictured in subfigure 4.6d using equations 4.9 and 4.10.

$$X = \cos\gamma * D \tag{4.9}$$

$$Y = \sin\gamma * D \tag{4.10}$$

Orientation and position calculation

The angle of the orientation is defined in reference to the camera frame and measured counter clockwise from the $X - Axis$ as shown in figure 4.7a in a range of $0 - 2\pi$.

Using the 3D position of the keypoints, the orientation of the detected person can be calculated as shown in figure 4.7b. Here the angle β can be calculated according to equation 4.13 using δX and δY from equations 4.11 and 4.12. $arctan2$ is used to distinguish the orientation of a person in the spectrum of a full rotation, however that means, that the resulting angle β has to be transformed back into the spectrum of $0 - 2\pi$ as shown in equation 4.14.

Subtracting this angle from 2π results in angle α and subtracting another 0.5π from that results in the angle θ shown in figure 4.7b according to equation 4.15.

$$\delta X = X_{right-shoulder} - X_{left-shoulder} \tag{4.11}$$

$$\delta Y = Y_{left-shoulder} - Y_{right-shoulder} \tag{4.12}$$

$$\beta = arctan2(\delta Y, \delta X) \tag{4.13}$$

$$\beta = \begin{cases} \beta, & \text{if } \beta \geq 0 \\ \beta + 2\pi, & \text{otherwise} \end{cases} \tag{4.14}$$

$$\theta = (\pi - \beta) - \frac{\pi}{2} = \frac{\pi}{2} - \beta \tag{4.15}$$

This method of finding the orientation of a person can just as well be applied to other keypoints located on either side of the detected person. In this work, the orientation given by the shoulders is always preferred. If the shoulders are not being detected the right and left hip joint is taken alternatively.

If neither, both shoulders nor both hip joints can be found the angle will default to zero and the KF will be updated without a measurement of θ as described in more detail in section 4.2.3. This however results in the state of the orientation and the corresponding velocity being not as reliable as the position, since it is likely that the person might be oriented in a 90° angle to the robot and thus only one shoulder and hip can be seen and detected.

The 2D position of a person is calculated with the relevant keypoints, which are neck, right and left shoulder and right and left hip joint. The resulting Position is simply set to the mean position of these five points (or less if some are not detected).

Occlusions may result in these points not being detected which will result in people not detected. This can be due to the person being too close to the robot or being occluded by objects in the scene.

2D map pose

To track the 2D pose (2D position and Orientation around the Z-axis) of all persons in the environment without being affected by the movement of the robot, the previously calculated pose that was relative to the tf_frame located at the camera has to be transformed into the fixed "map" tf_frame, such that they can be tracked there.

However, the detections are not always perfectly reliable, which leads to cases where the same person is detected multiple times. If these detections would be passed to the tracker without being filtered first, they would lead to additional tracklets.

To counteract this, a threshold of 0.5m has been determined, at which detections will be merged to their average 2D map pose. As 0.5m is the radius of the

intimate proxemic zone the assumption can be made, that no other person should be present within it in an industrial environment.

This allows to build a distance matrix between all detections that can be thresholded and afterwards be used to find and merge the duplicate using a basic average for X ,Y and θ respectively.

Tracking

Due to the noisy and partly unreliable measurements of the keypoints as well as to keep track of the humans in the surrounding environment of the robot a tracker has to be used.

As there is the possibility of multiple people in the environment, the tracker consists of two parts – an assignment part to ensure that the same person is tracked over time and a filtering part.

The assignment is done using Munkres' algorithm [80] while the filtering is performed using a KF [96] as explained in section 3.4.

Assignment

As highlighted previously the detections are assigned to the individual KFs using the Munkres' algorithm. As this algorithm computes the minimum cost assignment in a cost matrix, the tracking algorithm needs to be extended with case handling and a cost function.

The KF states consist of the X and Y position of a person, aswell as their orientation. In addition to these states the KF has the rate of these as additional states as described in section 4.2.3.

In this implementation the cost function $Cost_{distance}$ is represented as the euclidean distance between the X,Y position of the KF state and detection since the rates are likely to change quickly and the orientation measurement is not as reliable as the position measurement, as mentioned in section 4.2.3. Therefore the cost function can be expressed according to equation 4.16.

$$Cost_{distance} = \sqrt{(X_{detection} - X_{state})^2 + (Y_{detection} - Y_{state})^2} \tag{4.16}$$

The Munkres' algorithm returns the best combination of an $n \times m$ matrix. Therefore, it is possible that new detections will be assigned to tracklets, even though they are unreasonably far away therefore, the entire assignment would be skewed. Hence after generating the distance matrix D the minimum cost assignment is compared to a configurable threshold T_D. If the minimum cost of the detection is higher than T_D, the detection is removed from the distance matrix and a new tracklet is initialised for the detection. In this project $T_D = 3$m Afterwards the Munkres' algorithm can be run to generate the assignment combination with

the lowest total distance between detections and tracklets. Which leads to the following three cases:

1. The amount of detections and tracklets is equal: D is a square matrix and all detections will have an assignment to a tracklet. Therefore the detections can be directly set as a new measurement for the existing tracklets.
2. The amount of detections is higher than the amount of tracklets: All assigned detections can be set as a new measurement for their corresponding tracklet. Unassigned detections are initialized as new tracklets.
3. The amount of detections is lower than the amount of tracklets: All detections are assigned to tracklets. The remaining tracklets cannot be updated.

In addition to the cases above an additional constraint is necessary to remove old tracklets, hence the Threshold T_T is introduced which describes the amount of time allowed a tracklet is kept after the last update. If the timestamp of a tracklet surpasses T_T it can be removed from the list of tracklets. In this project $T_T = 5$s.

Filtering

The tracklets of this multi person tracker implementation are in the form of individual KFs as explained in section 3.4.2. However, the procedure of KFs can be applied to numerous cases and therefore needs to be designed to handle the specific case of people tracking in the case of Nav2CAN.

One of the requirements of the tracker in this project is to have both, 2D position, as well as the orientation of all people in the field of view of the robot.

Therefore the states of the KF will be set to those. When considering moving people however, the KF would stay still during the prediction step, if the rate of the position and therefore the velocity is not tracked. Hence, the state vector is enhanced with the rate of all three required states and can be expressed as shown in equation 4.17.

$$\hat{x} = \begin{bmatrix} X \\ Y \\ \theta \\ \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} \tag{4.17}$$

Both, X and Y and their velocities that are modelled with a constant velocity model. Therefore, they are linear unbound continuous variables and hence can be tracked with a KF.

However, the orientation has a discontinuous phase. Therefore, if the angle would be tracked based on an input angle in a range of $0 - 2\pi$, the angle would start to decrease after the measured person has turned past the limit of 2π instead of increasing linearly. This can be counteracted by "unwrapping" [97] the measured angle and therefore adding an offset of $k * 2\pi$ such that the difference of the state and the measured angle is never larger than π before updating the KF with the measurement. Hence the angle can be tracked as being quasi continuous.

Since the robot in this work has only a limited FOV, it is likely, that the robot looses vision to the person while it is in their vicinity. Thus it is important, that the position of these humans will be predicted, even if they are not seen. It is however unlikely, that humans behave based on a constant velocity model. Especially in an indoor environment it is unlikely, that humans can keep moving without any boundaries. Equally unlikely is that they will keep spinning in circles while not being seen. Thus, a decay is applied to the velocity states. This leads to the human position being predicted to slow down as soon as they are not seen anymore. This decay is implemented according to equation 4.18 where *decay* is a scalar that is being scaled by the predict frequency of the KF. In this project $decay = 0.9$ which allows for a slow decay.

$$\hat{x} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & decay * dt & 0 & 0 \\ 0 & 0 & 0 & 0 & decay * dt & 0 \\ 0 & 0 & 0 & 0 & 0 & decay * dt \end{bmatrix} \cdot \hat{x} \tag{4.18}$$

As mentioned in section 4.2.3, the orientation cannot alway be determined since there are cases where not enough keypoints can be seen. In this case it is important, that the KF is updated using an alternative measurement mapping matrix H and measurement noise covariance matrix R, such that the theta state is not affected by the unreliable measurement.

4.2.4 Publishing

The people detection module results in being the conversion between n streams of RealSense cameras to a list of KFs. Since this module is packaged in the form of a ROS2 node, a custom ROS2 message, is used for the communication to the modules of the software stack that rely on the information.

This custom message is a re-implementation of the ROS *package people_msgs* [98]. This package is however, not publicly available for ROS2 Humble and its message types "people" and "person" have been implemented as a ROS2 message and packaged into the package "multi_person_tracker_interfaces".

Here the People message contains the information shown in table 4.1, whereas the person message is represented by the structure shown in table 4.2. The structure of these messages enables the use of alternative algorithms for detection and tracking as the output can be converted to this format.

Table 4.1: people ROS2 message

std_msgs/Header	header
multi_person_tracker_interfaces/Person[]	people

Table 4.2: person ROS2 message

geometry_msgs/Point	position
geometry_msgs/Point	velocity
float64	reliability
string[]	tagnames
string[]	tags

4.3 Context module

This module includes the necessary nodes to detect the context of a scene and represent it in a way that is interpretable by conventional costmap based planners available for Nav2.

4.3.1 Social map generator

The input for the interaction detection is designed to be a grey scale image such that proxemics, as explained in section 2.1.1, are included in the input as further explained in section 4.3.2.

Therefore a node is required that generates this output from the poses of all people in the environment. Here the egg-shaped proxemic model introduced in section 2.1.1 is taken as a starting point. The computation of these values are by Gines Clavero et al. [99] as also shown in equations 4.19.

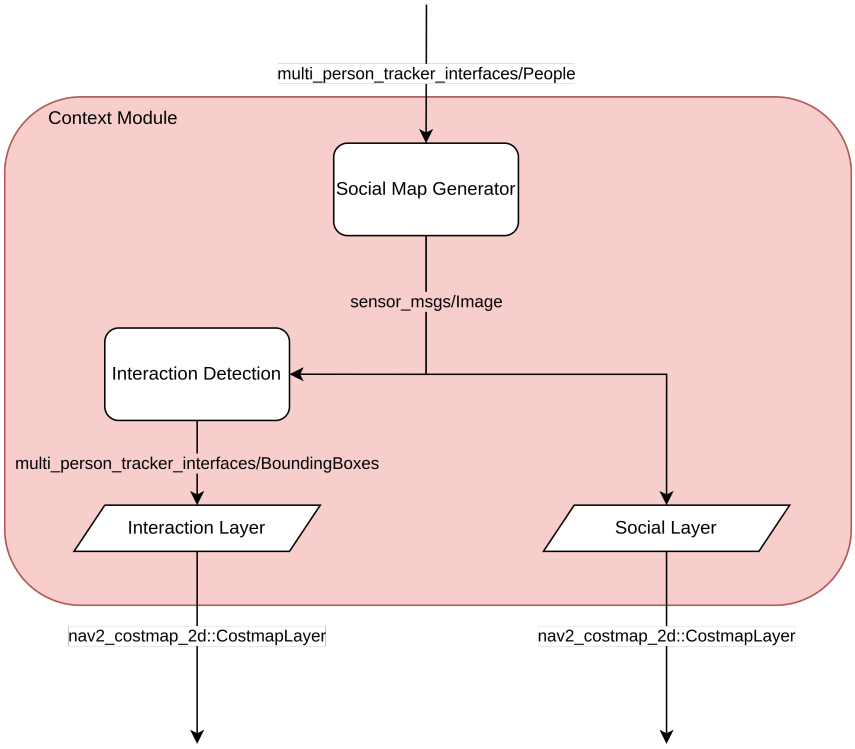


Figure 4.8: Diagram of the context module. Rounded rectangles represents ROS2 nodes, whereas parallelograms represents costmap plugins for the Nav2 navigation stack. Connections between module components represent the relevant messages

$$value(x, y, theta) = e^{-(A(x-x_i)^2+2B(x-x_i)(y-y_i)+C(y-y_i)^2)} \tag{4.19}$$

Where:

$$A = \frac{\cos(\theta)^2}{2*\theta^2} + \frac{\sin(\theta)^2}{2*\theta_{side}^2}$$

$$B = \frac{\sin(2\theta)^2}{4*\theta^2} - \frac{\sin(2\theta)^2}{4*\theta_{side}^2}$$

$$C = \frac{\sin(\theta)^2}{2*\theta^2} + \frac{\cos(\theta)^2}{2*\theta_{side}^2}$$

θ_{side} is the variance to left and right

θ is variance to the front or back depending on the pixel coordinate

The resulting value distribution is then thresholded according to equation 4.20 resulting in the cost distribution. Where the Gaussian function reflects a Gaussian distribution at the origin with the variance of θ_{side} and $value$ is the one calculated according to 4.19 which results in the cost distribution shown in red in figure 4.10.

The reason for the thresholded cost instead of the smooth asymmetric Gaussian is to define values for the different proxemic zones instead of smoothing their transition. Here the intimate zone gets a very high value of 253 so the robot can never advance into that zone. The personal zone however, is divided into a fixed cost section and a decaying section. The reasoning for this is that the advancing of the robot into the personal zone should be discouraged by increasing the cost at the edge of the zone. As the robot might be required to enter the personal zone of a person in order to interact with them, the cost is not further increased within that zone, such that robot can position itself anywhere within it. This is meant to force the robot to take the shortest possible path through the personal zone to its goal. An example of the generated map can be seen in figure 4.9.

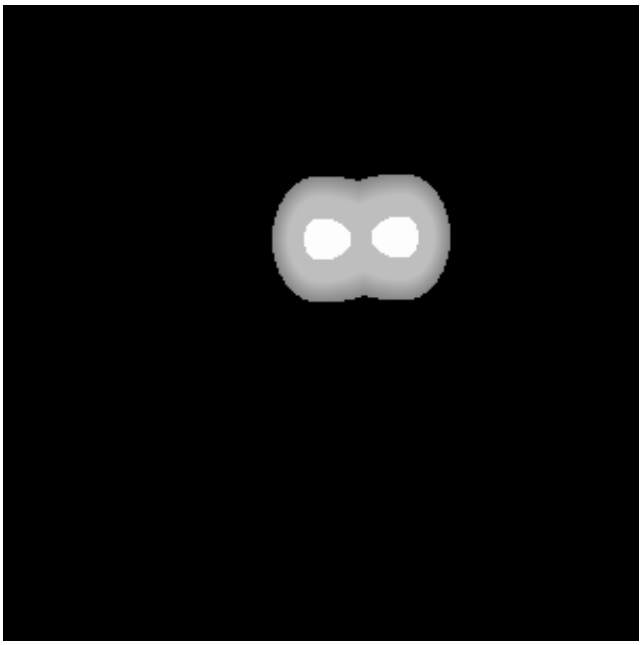


Figure 4.9: An example of a generated social map of two people facing each other with overlapping social zones. The centre of each person has a higher cost value which lowers as the colour turns from white to grey.

To prevent the people flooding the entire costmap with low cost values as the Gaussian decays, the distribution is clipped off at the edge of the personal zone such that the robot is free in its movement.

$$cost = \begin{cases} maxcost, & \text{if value > Gaussian(0.5)} \\ maxcost * Gaussian(1.0), & \text{if value > Gaussian(1.0)} \\ value * maxcost, & \text{if value > Gaussian(1.5)} \\ 0, & \text{otherwise} \end{cases} \tag{4.20}$$

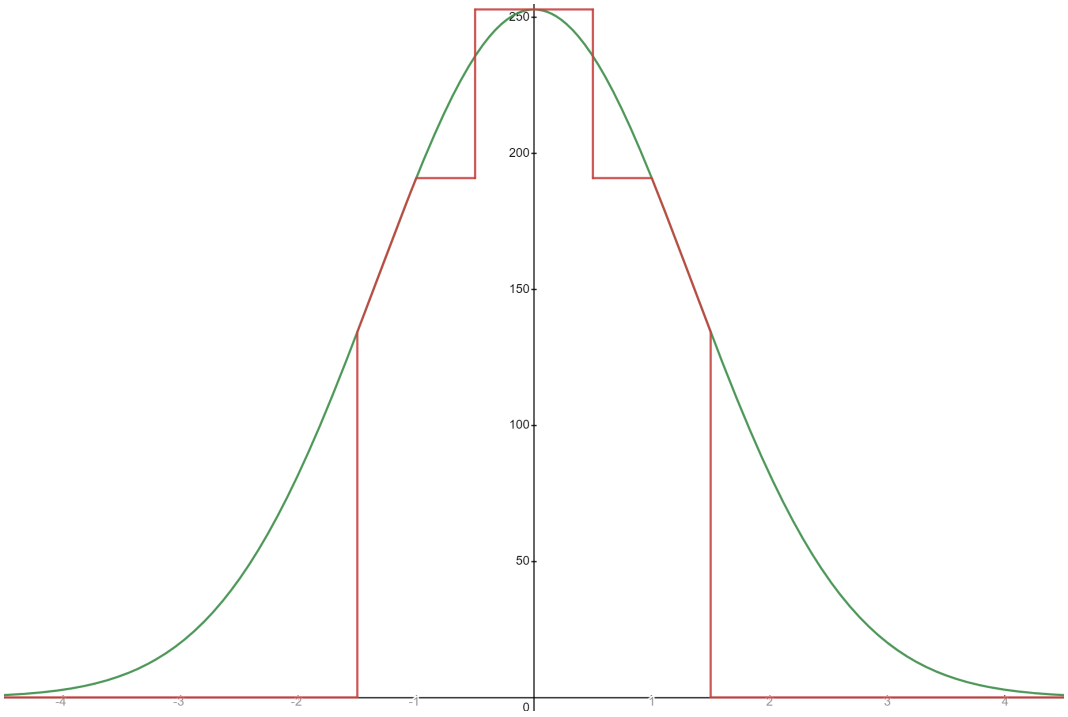


Figure 4.10: Slice along the y-axis of the cost distribution. Red is the thresholded cost, green the values of the original cost distribution. The y-axis represents the cost value, whereas the x-axis represents the distance in meters from the centre of the person

The social map has the task of representing the people, at the time of their recording, relative to the current position of the robot. This introduces a time requirement since the runtime of the detector pipeline and or the update frequency of the costmap introduce delays during which the robot might be moving.

The social map is centred at the robots position but is not rotated but instead aligned to the *map* frame. This results in the benefit, that interactions in the map are not moving drastically when the robot rotates in order to avoid the people in the environment. With good localisation, this simplifies the combination with the local costmap as the local costmap is aligned to the *odom* frame. Therefore, both "images" are already well aligned. With robot platforms that have bad odometry the *map* and *odom* frame might be shifted and or rotated in relation to each other requiring a transformation of the image, however in the case of this work the localisation and odometry is good enough for the frames being well aligned.

Due to the alignment of the social map to the map frame, only the translational component of the tf_transformation between the robots *base_link* tf_frame and the tf_frame the people where detected is required.

However, as mentioned earlier, the runtime of the individual nodes e.g. detector and tracker will introduce a delay of the information and since the robot

or the environment is non-static this would translate into a physical offset which might hinder navigation.

Therefore, passing on a timestamp from the captured image, used for detecting people and interactions, is necessary. This ensures that the detections can be referenced at the time of capture and thereby located correctly relative to the robot at current time.

As the proxemic zone is a grid of cost values, it is computationally intensive to calculate. Thus the cost distribution is calculated during initialisation of the node and then rotated and copied onto the social map according to the pose of each detected person.

4.3.2 Interaction detection

The interaction detection developed in this project uses the YOLOv7 architecture, as described in section 3.5. Here the *tiny* version of the architecture is used, as it is developed for use on edge devices.

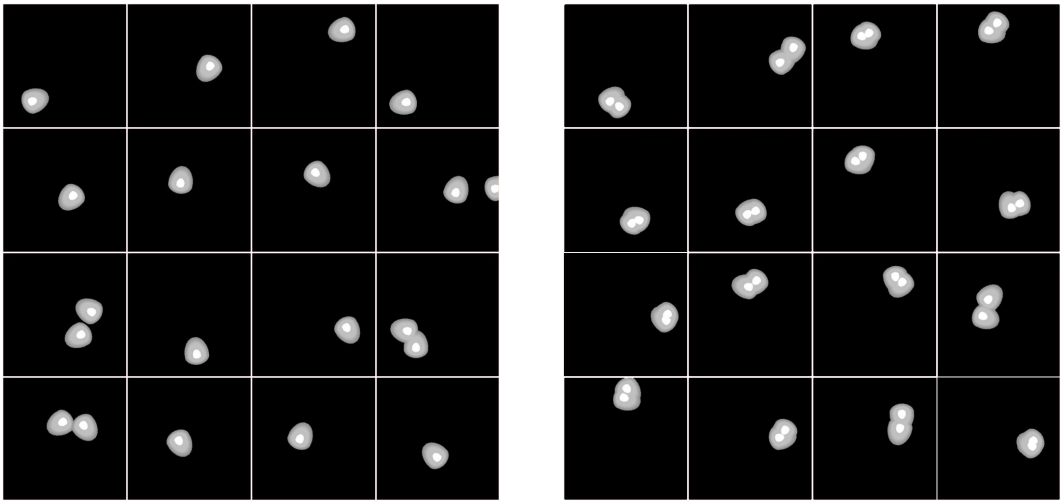
The model is trained using data generated from the people detector and social costmap generator. The data is augmented to cover the whole map, as the data was generated using a stationary setup which limited the FOV of the camera. The setup is similar to the one shown in figure 4.4. This means that the data is augmented to rotate the data around the centre of the map, where the robot is located, to enable detection in the full map. with a resolution of 1 pixel corresponding to 5cm in the map, a map of $300px$ by $300px$ represents a map of 15m by 15m. This means that a detection can be made within 7.5m of the robot, further than the requirements described in section 2.5.1.

Due to the amount of images gathered from different distances with both interactions and no interaction, the dataset amounts to more than 500,000 images after augmentation. As the images were manually divided into the classes interaction and no interaction prior to augmentation, the data can be automatically annotated using the features of the social zones. Figure 4.11 shows examples of the images in the dataset.

The trained network is constructed as a ROS2 node that subscribes to the generated social map and detects interactions within the map. The detections are then re-published as a message of type BoundingBox as shown in table 4.3 such that the detected interactions can be represented as explained in section 4.3.4.

Table 4.3: BoundingBox ROS2 message

std_msgs/Header	header
float64	center_x
float64	center_y
float64	width
float64	height



(a) Examples of images classified as people not interacting. Here, the context is important as two people can be standing close to each other in a short while but are simply passing each other while moving.

(b) Examples of images of people interacting. Here, the context is important as two people must stay close to each other for a longer time for it to be an interaction, to distinguish from walking past each other.

Figure 4.11: Examples of the images used in the dataset from both classes of interaction and no interaction.

4.3.3 Social Layer

This layer makes use of the already computed social map from the social map generator introduced in section 4.3.1. As briefly mentioned in section 4.3.1, a transformation in the case of unreliable odometry would be required. As both, the costmap and the social map are image like grids the social map can be overlaid and thereby turned into a costmap with the pixel values representing the cost for traversing. Figure 4.12 shows the visualisation of the costmap for two people facing each other.

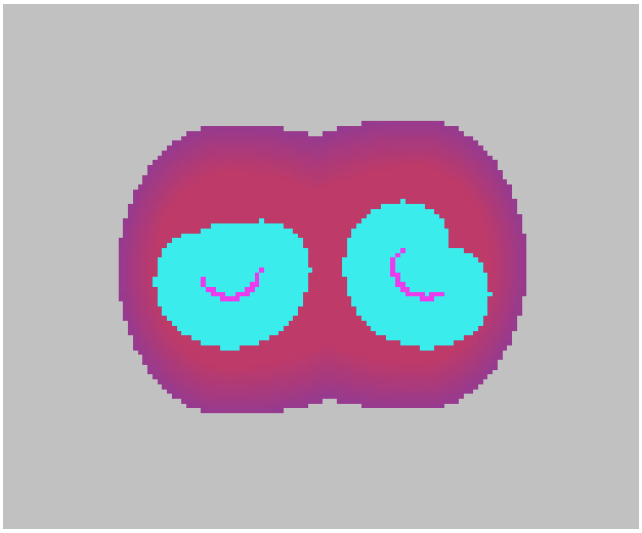


Figure 4.12: An example of the costmap of two overlapping social zones visualised RViz. The turquoise represents the inflation from the laser scan (pink half circles) of the two people with the added social zone in read and purple around them. The changing colour represents the change in cost values as the distance grows from the center of the person.

4.3.4 Interaction Layer

Similar to the social costmap layer, the interaction layer is also implemented as a costmap_2d layer and therefore a plugin that can be added to any Nav2 based navigation stack. The task of this layer is to draw elliptical cost distributions based on the bounding boxes detected by the interaction pipeline introduced in section 4.3.2. The shape of the elliptical zone is an approximation of the P space, described in section 2.1.1. As it is unknown whether this layer is implemented in the global, local or both costmaps, the coordinates of the center have to be transformed into the global tf_frame of the affected costmap and the current time. This avoids a swerving cost distribution that might largely hinder path planning and navigation.

From the received bounding box, the ellipse is drawn into the map layer using the spcified value in the Nav2 parameter setup. In this project this value is set high to highly discourage the planner from entering an interaction zone. The generated layer is then combined with any other layer in the costmap to ensure maximum cost at every cell. An example of the resulting costmap can be seen in figure 4.13.

Figure 4.13: An example of the costmap of an interaction zone of two people, visualised in RViz. The location of the two people from the laser scan is shown with the the two half circles in purple while the elliptical interaction zone is shown in turquoise. The constant colour represents the constant cost of the interaction zones.

4.4 Nav2

The navigation stack of Nav2CAN, be it in simulation or with a real robot, is based on the default configuration stack introduced in the *nav2_bringup* repository [100]. This configuration is shortly introduced in this section along with problems that occurred during implementation.

4.4.1 Configuration

This subsection will shortly introduce the Nav2 configuration of Nav2CAN, as shown in figure 4.14.

The planner of the navigation stack in this project is the default planner *nav2_navfn_planner/NavfnPlanner*. This planner is then accompanied with the smoother *nav2_smoother::SimpleSmoother*, in order to remove edges caused by the grid based path finding techniques and to improve path feasibility.

This smoothed path is then send to the controller which in the case of Nav2CAN is represented through a *FollowPath* behaviour for the previously mentioned smoothed path. The path following is handled by the controller plugin *dwb_core::DWBLocalPlanner*.

Meanwhile, the plugins *nav2_controller::SimpleProgressChecker* and *nav2_controller::SimpleGoalChecker* are monitoring progress and distance to the goal. The resulting output of the controller are the command velocities, that however are not directly sent to the robot but instead first to the velocity smoother

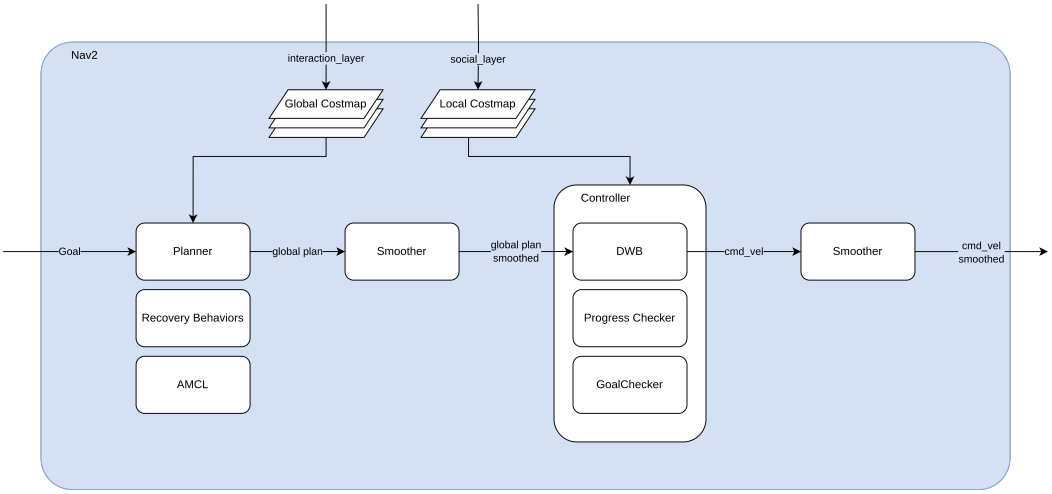


Figure 4.14: Block diagram of the Nav2 navigation stack showing the interconnection between the individual nodes. Topics to and from the individual nodes are not drawn.

plugin *nav2_velocity_smoother*, in order to reduce acceleration and jerky movements to protect the hardware of the platform.

In case of failed planning or controlling, multiple recovery behaviours are added to the behaviour server. These are:

- *nav2_behaviors/Spin*
- *nav2_behaviors/BackUp*
- *nav2_behaviors/DriveOnHeading*
- *nav2_behaviors/Wait*
- *nav2_behaviors/AssistedTeleop*

The two costmap layers discussed in section 4.3.3 and 4.3.4 can now be used as plugins for the costmaps of the navigation stack.

The interaction layer is only added to the global costmap such that movement through interactions can be discouraged by using an ellipse for the interaction with a high cost. The resulting layers are therefore as follows:

- interaction layer
- static layer
- obstacle layer
- inflation layer

The social layer can be added to the local costmap such that the controller can steer the robot in accordance to the defined proxemic constraints. This results in the following layer configuration:

- social layer
- static layer
- obstacle layer
- inflation layer

Finally the localisation of the navigation stack is represented by the adaptive Monte-Carlo localisation based node *AMCL*. This node is localising a robot using its odometry, a scan of the environment along with a map of it.

For the simulation part, the parameters from *nav2_bringup* have been modified in accordance with the parameter found in the *CoHAN Planner* setup, such that it is compatible with the simulated PR2 robot [101]. This means that the robot will only be sufficiently tuned for navigation as this is not the focus of the project. Furthermore, due to differences between navigation stacks in ROS1 and ROS2, the performance of the two systems may differ in performance and visual feedback in RViz.

The same navigation stack is deployed in order to be used with the MiR platform. However, since the internal software is highly tuned for the platform the internal localisation is used. This means, that the platform not only publishes the required messages defined in section 2.5.1 but now also the internal map along with the *map -> odom* tf_transform.

4.5 CoHAN simulation

The CoHAN Planner [75] was tested in simulation using MORSE [102] but also provides a simulation in *Stage* [103]. Stage is a robot simulator that provides computationally cheap methods for simulating multiple agents in an environment. Furthermore, Stage provides the user with multiple devices for simple integration of methods. A Stage world is defined by populating a world from an image with actors in a script. Due to the simplicity of the simulation it can run on lower-tier hardware, compared to graphics intensive simulators. Figure 4.15 shows the world used from Stage.

Multiple different configurations of the modified world are presented in chapter 5. This world is adapted from a world used by CoHAN simulation.

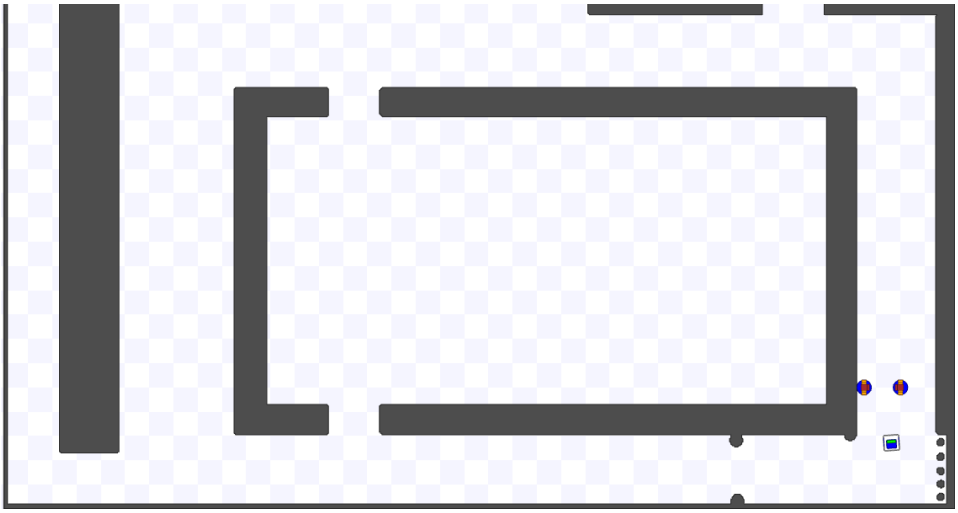


Figure 4.15: An example of the test environment. The environment contains walls (grey), a simulated robot (white, blue and green box) and two humans (blue circles). The location of people are published on the ROS network to be detected by the robot when planning movement.

4.6 ROS1 and ROS2 bridging

As described in section 3.1, there are multiple differences between ROS1 and ROS2. Thus bridging between the different versions is needed, when packages are not yet available for ROS2

4.6.1 Simulation

As the simulation used by the CoHAN planner is developed for ROS Melodic, the environment is kept the same for the comparison with the systems developed in this project. Therefore, a bridge between ROS1 and ROS2 is used to pass the necessary messages between the two. This is done using *ROS1_bridge* [104] on the computer running the simulation. Using the *parameter_bridge* enables the selection of topics to be bridged between ROS1 and ROS2. By choosing which topics are passed between the two versions of ROS it is possible to limit the amount of bandwidth to only the needed topics.

This means that the ROS1 simulation is run completely on the simulating computer with the navigation stack provided from the CoHAN planner. For the comparison only the simulation is run in ROS1 with the robot and the different messages from the simulation is passed through the bridge to the ROS2 network. Here the robot controller is running where the navigation stack used in this project handles the control of the robot. The simulation provides the sensor messages of the robot, that is odometry and laser scans, and receives in exchange command velocities.

Furthermore, using the *ROS1_bridge* enables running measurements in ROS2 during runs in both systems. This means that nodes for gathering data for travelled distance and distance between robot and people can be run using ROS2 while running the CoHAN planner in ROS1. These nodes use the topics published by the Stage simulation, bridged from ROS1 to ROS2 and then collected for data handling. Using the same method for gathering data in both scenarios ensures that the values are comparable.

4.6.2 MiR100

Bridging is also required in the case of the MiR100 platform used in Nav2CAN. This is caused by the platform internally using ROS1 as the middleware. However, as the internal PC of the platform cannot be accessed the data is extracted using the *mir_driver* that is part of the *mir_robot* project developed by Martin Günther [105]. The *mir_driver* is a reverse ROS bridge for platforms developed by *MiR*. This allows the direct access of the sensor data and internal information of the platform such that it can be controlled from an external controller while disregarding parts of the internal software stack.

However, the *mir_driver* is not yet available for ROS2 and therefore a *ros1_bridge* that bridges between the *mir_driver*, running on ROS1, and ROS2 is required.

As explained in section 4.4 the decision was made to use the internal localisation algorithm of the *MiR100* since it has been tuned by the manufacturer in order to achieve the best performance. This means however, that the platform needs to publish the *tf_transform*, as well as the internal map.

5 - Results

This chapter deals with running Nav2CAN in different setups, both simulation and on a real robot. This is to showcase how the different modules developed in this project perform in different scenarios. This means that some configurations will change, depending on the used setup as the underlying robot will differ.

These tests are limited to show a proof of concept and will therefore not address every edge case but rather the needed aspects to determine the success of the developed modules in accordance with the requirements described in section 2.5.1. This means that some parts will be tested on their own while others will be tested in connection with others. This is due to the modular development where some parts are developed for individual use.

As described in chapter 4, the setup is tested on both a simulated environment as well as a physical robot setup. It is however important to note that both of these setups are using ROS1 and therefore need to be bridged to ROS2, as described in section 4.6.

5.1 Module timings

In order to determine the runtime of Nav2CAN, it is important to know the runtime of each module in the system. To do so, a timing has been implemented into each of the nodes to calculate the time it takes to run the node once. The resulting runtimes are an average of more than 1,000 individual timings. Accumulating the node timings gives a good approximation of how long it takes for one image from the RealSense to turn into interaction zones. Table 5.1 shows the different average times of the nodes. Here it can be seen that the Interaction Detection and the Social Map Generator are the slowest modules with 35.79ms and 36.75ms, respectively. In order to measure the runtimes reliably, all timings have been performed while all three modules of Nav2CAN are running. Furthermore, as the runtime of many nodes can depend on the amount of people within the environment said timings have been capture in normal conditions, here defined as a scene with zero, one and two people.

Table 5.1: Average runtime of each module in Nav2CAN

Module	Average runtime (in ms)
Multi Person Tracker	14.02
Social Map Generator	36.75
Social Layer	1.91
Interaction Detection	35.79
Interaction Layer	0.19
Accumulated Time	88.66

5.2 CoHAN comparison

As stated in the CoHAN paper [75]:

> *Since the local planner runs a computationally expensive optimisation in each control loop, extending the planning beyond two humans does not yield real-time control of the robot. Hence we restricted our human planning to the two nearest humans.*

This means that the CoHAN planner is not developed for multiple people but will handle the two nearest. It is therefore of interest to compare the performance of the planner with Nav2CAN, the system developed in this project.

To compare the performance of this project with that of CoHAN, a simulation is carried out using Stage, as described in section 4.5. Here the simulation is based on the setup from the CoHAN planner [106]. This enables a direct comparison between the two systems by recording the same metrics. For this comparison the metrics will be:

- Length of the planned path.
- Minimum distance to a person during movement.
- Disturbance of social zones.

The environment will contain different areas with different sizes populated with different amounts of people. The people will be standing in different configurations for interaction. An example of the environment from the Stage simulation can be seen in figure 5.1. The purpose is to determine how the two planners handle interactions between people and how the movement of the robot may disrupt the interaction zones and thereby the potential comfort of the people.

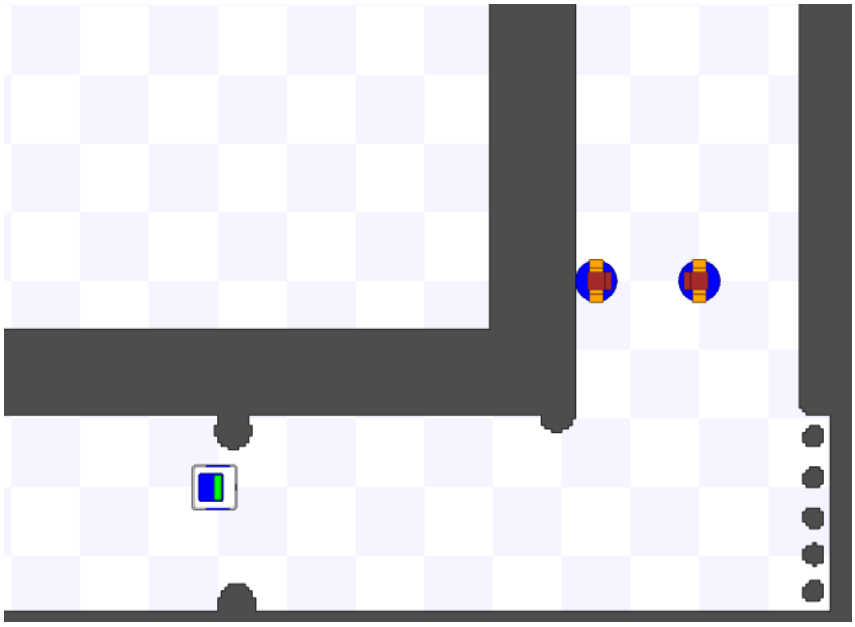


Figure 5.1: An example of the test environment. The environment contains walls (grey), a simulated robot (white, blue and green box) and two humans (blue circles). The location of people are published on the ROS network to be detected by the robot when planning movement.

5.2.1 Publishing people on the ROS2 network

As the humans used in the Stage simulation are published for the CoHAN planner it is important to ensure that the people are published in a usable manner when bridged to the ROS2 network. Due to the difference in message type between ROS1 and ROS2 a publishing node is developed which ensures that the location of people in the simulation are available for nodes running on ROS2.

This node subscribes to the topic published for each individual person in the simulation and creates a Person message for each which are published as a list of People. This list can then be used in the different nodes to ensure that the Nav2CAN system can detect interactions. As mentioned in section 4.2 a similar node could be developed to translate the output for an alternative people detector.

5.2.2 Comparison scenarios

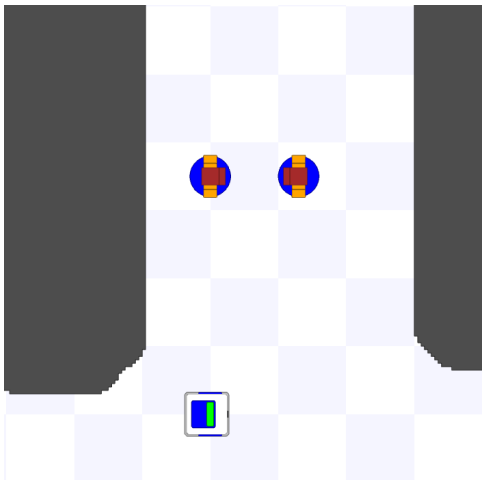
As it can be seen from the simulation image in figure 5.1, the environment is simple and serves the purpose of comparing navigation around humans. The different comparison scenarios will contain between two and four humans in various constellations and with different starting and goal positions of the robot. Each scenario will be run between five and ten times times with both methods and the average travelled distance and minimum distance to a person will be compared

for each scenario. Five types of scenarios will be used and two starting and goal positions for each scenario type to ensure that the robot handles the scenarios similarly from both sides. The type of scenarios are as follows:

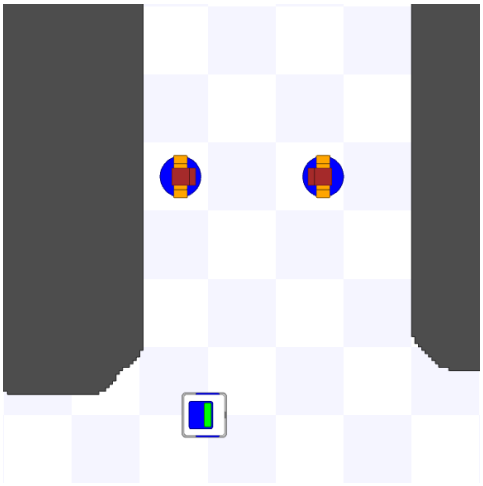
1. Two people standing face to face in the middle of a 4m wide area at personal distance, as described in section 2.1.1.
2. Two people standing face to face in the middle of a 4m wide area at social distance.
3. A group of three people standing in a 4m wide area in an F-formation, as described in section 2.1.1.
4. A group of four people standing in a 4m wide area in an F-formation.
5. Two people standing at social distance in an open room.

Figure 5.2 shows a version of each of the comparison setups. Here the robot must move from one side of the interaction to the other.

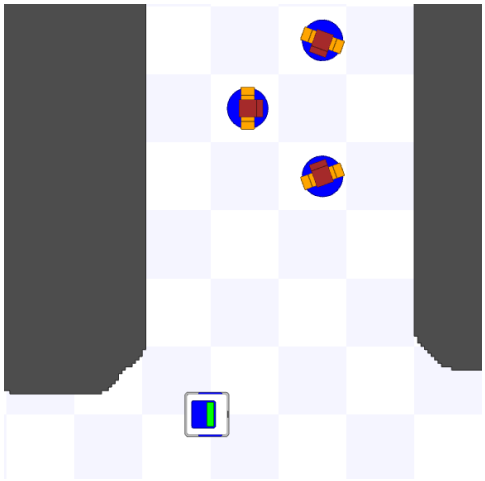
The following sections describes the outcome of the different comparisons to determine different situations that are better suited for one system or the other. Tables 5.2 and 5.3 shows the relevant values collected during runs to determine the difference between the two systems. Furthermore, the following sections describe how different social zones were or were not disturbed during the runs to determine any differences between the systems for interaction detection and handling.



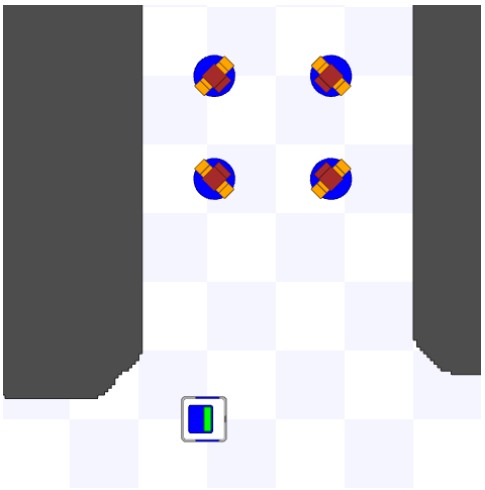
(a) Two people standing in a corridor to the side.



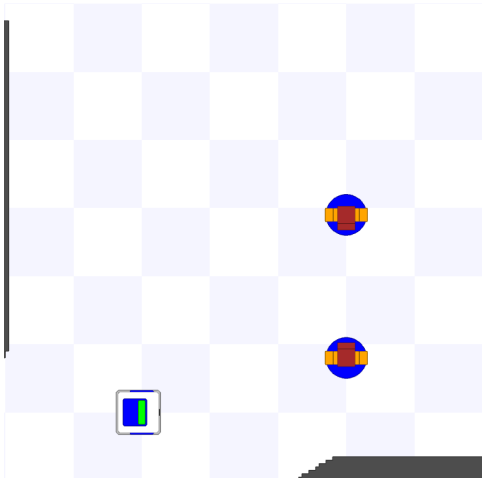
(b) Two people standing in the middle of a corridor.



(c) A group of three people standing to the side in a corridor.



(d) A group of four people standing in a wide corridor.



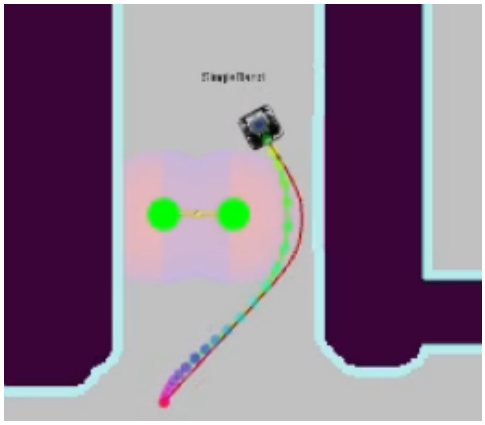
(e) Two people standing in an open room.

Figure 5.2: Different environment setups for comparison between CoHAN and Nav2CAN. The robot indicates starting position and must plan a path to move on the other side of the people. Each square represents a space of 1m $\times$ 1m.

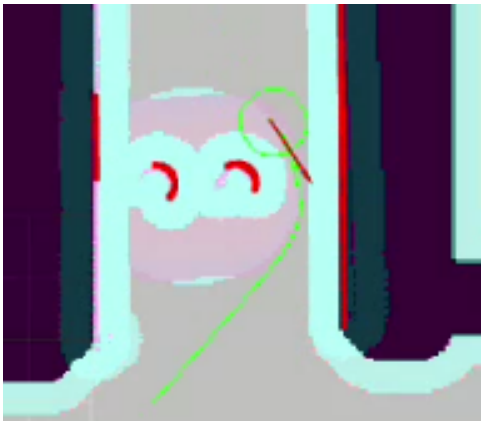
Two people at personal distance

The first scenario determines how the two systems handle two people standing at personal distance while interacting in a corridor. Here, there is room for the robot behind one person which can enable the robot to move behind said person while avoiding the humans as obstacles.

As can be seen from figure 5.3 the paths chosen by both systems are similar. Both systems plan a path behind one person, CoHAN at a distance of 0.89m and Nav2CAN at 0.94m. This means that when moving behind a person, the two systems perform similarly. It can be seen in the figure that the CoHAN local planner tends to cut off slightly more of the distance to the person which results in the closer distance. This also results in a slight shorter travelled distance of the robot with 9.44m by CoHAN and 9.58m by Nav2CAN.



(a) Path planned by CoHAN. The red line shows the global path while the dotted line shows the local path. The two humans are represented as green circles with their social zones (as used by CoHAN) are shown.



(b) Path planned by Nav2CAN. The green line shows the global path of the global path which is followed closely by the controller. The pink oval shows the interaction zone in the global costmap while the people are only shown as red laser scans.

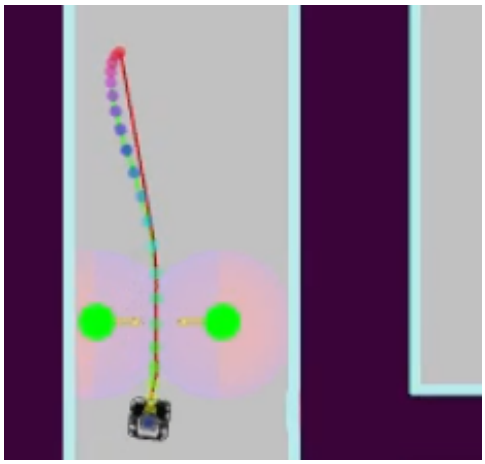
Figure 5.3: Scenario 1 - two people standing at personal distance in a corridor.

Two people at social distance

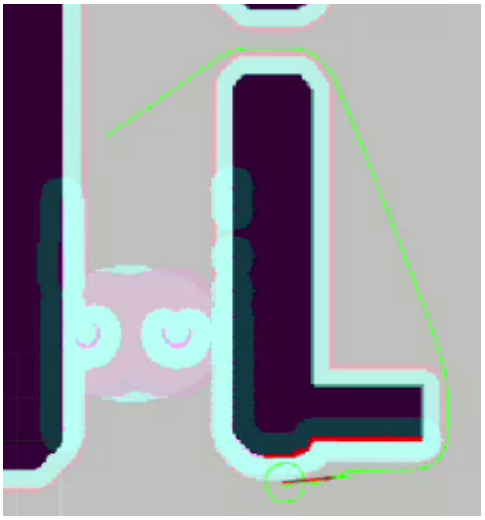
The second scenario highlight the difference in how the two system perform when the interaction is at social distance and there is no longer room to traverse behind a person.

Figure 5.4 shows the different planned paths from the two systems. CoHAN plans to travel between the two persons, effectively disrupting the interaction. Nav2CAN plans a route to completely avoid traversing between the two people. This results in CoHAN getting within a distance of 1.00m of a person, effectively disrupting the personal space in front of the person. On the other side, Nav2CAN

chooses a path that results in a distance to the people not getting closer than 3.50m to a person. This results in a travelled distance of 8.22m by CoHAN and 23.50m by Nav2CAN.



(a) Path planned by CoHAN. The red line shows the global path while the dotted line shows the local path. The two humans are represented as green circles with their social zones (as used by CoHAN) are shown.



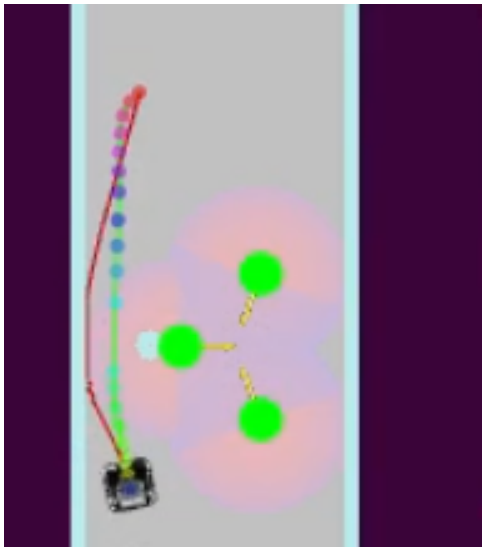
(b) Path planned by Nav2CAN. The green line shows the global path of the global path which is followed closely by a local planner. The pink oval shows the interaction zone in the global costmap while the people are only shown as red laser scans.

Figure 5.4: Scenario 2 - two people standing at social distance in a corridor.

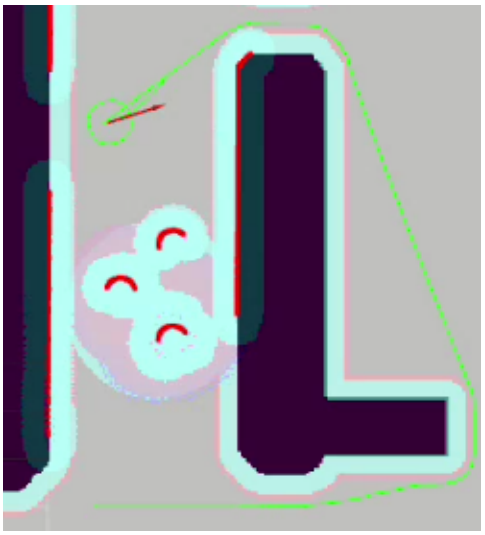
A group of three people

In the third scenario a group of three people are standing in the corridor, facing each other. This shows how the systems handle interactions of multiple people. Due to the layout of the setup, the robot has a possibility to traverse behind the people.

Figure 5.5 shows the planned path of both systems. Here it can be seen that CoHAN again tries to move behind a person which results in a distance of 0.76m to that person. This results in a travelled distance of 8.49m using the CoHAN planner. The Nav2CAN regards the interaction as filling the whole corridor and therefore chooses to plan away from the corridor, similar to scenario 2. This results in a distance to the people of 2.88m with a travelled distance of 23.54m.



(a) Path planned by CoHAN. The red line shows the global path while the dotted line shows the local path. The two humans are represented as green circles with their social zones (as used by CoHAN) are shown.



(b) Path planned by Nav2CAN. The green line shows the global path of the global path which is followed closely by a local planner. The pink oval shows the interaction zone in the global costmap while the people are only shown as red laser scans.

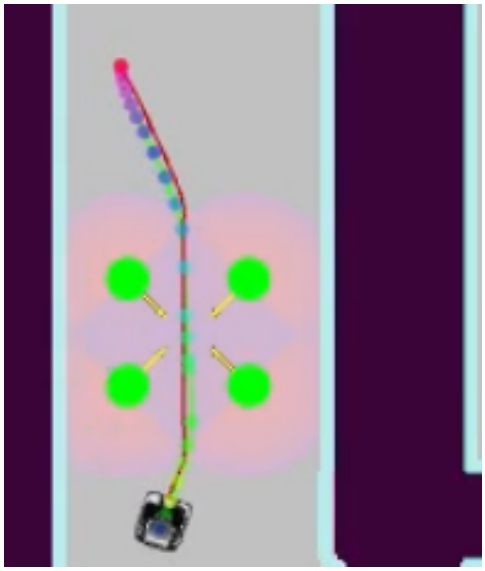
Figure 5.5: Scenarios 3 - three people standing in a corridor.

A group of four people

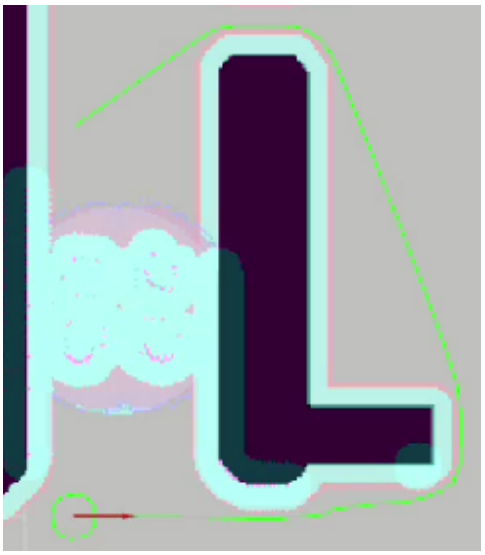
In the fourth scenario a larger group of four people is standing in the corridor. Here there is no room to traverse behind people due to the formation of the group.

As can be seen in figure 5.6, the CoHAN planner plans the path in the middle of the group, disrupting the interaction. This results in a distance to the closest person of 0.72m while travelling a total of 8.46m. This results in moving into the personal space of all of the people.

Nav2CAN again calculates a path that does not disturb the social interaction and therefore takes another path as can be seen in the figure. This results in a minimum distance to people of 3.00m with a travelled distance of 23.32m.



(a) Path planned by CoHAN. The red line shows the global path while the dotted line shows the local path. The two humans are represented as green circles with their social zones (as used by CoHAN) are shown.



(b) Path planned by Nav2CAN. The green line shows the global path of the global path which is followed closely by a local planner. The pink oval shows the interaction zone in the global costmap while the people are only shown as red laser scans.

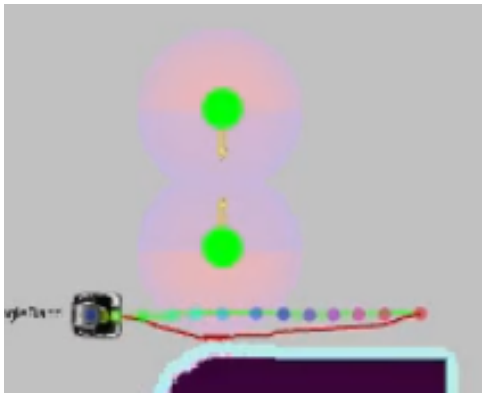
Figure 5.6: Scenarios 4 - four people standing in a corridor

Two people in an open room.

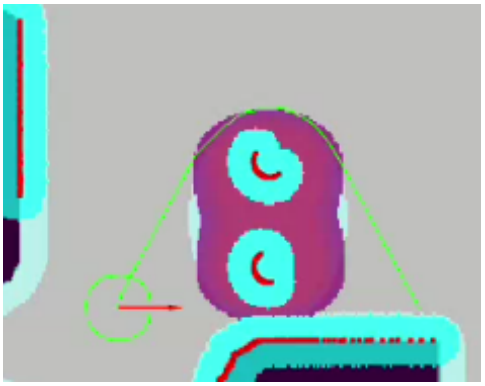
The fifth scenario determines how the planner perform when two people are standing in an open room, leaving enough room to move further away from the people. Here one person is standing close to the wall while the other has more than sufficient room for passing behind them.

The path planned by the two systems can be seen in figure 5.7. Here it can be seen that the CoHAN planner moves the robot behind the person close to the wall, as it deems enough space available. This results in a distance of 0.84m between robot and person. When some noise is introduced in the localisation of the robot, the robot will in some instances try to travel further into the open room resulting in a longer distance. This results in a average travelled distance of 7.73m for the CoHAN planner in the fifth scenario.

The path planned by Nav2CAN directs the robot around the back of the person in the room and thereby keeps a further distance to the people. The distance is in regards to the social zone and therefore is 1.02m when closest. This leads to a travelled distance of 11.77m on average.



(a) Path planned by CoHAN. The red line shows the global path while the dotted line shows the local path. The two humans are represented as green circles with their social zones (as used by CoHAN) are shown.



(b) Path planned by Nav2CAN. The green line shows the global path of the global path which is followed closely by a local planner. The pink oval shows the interaction zone in the global costmap while the people are only shown as red laser scans.

Figure 5.7: Scenarios 5 - two people standing at social distance in an open room.

Table 5.2: Minimum distance (in meters) between person centre and robot centre during each scenario.

Navigation system	CoHAN	Nav2CAN
Scenario		
2 people - personal distance	0.89	0.94
2 people - social distance	1.00	3.50
3 people in a group	0.76	2.88
4 people in a group	0.72	3.00
2 people in an open room	0.84	1.02

Table 5.3: Average distance traveled (in meters) between person centre and robot centre during each scenario.

Navigation system	CoHAN	Nav2CAN
Scenario		
2 people - personal distance	9.44	9.58
2 people - social distance	8.22	23.50
3 people in a group	8.49	23.54
4 people in a group	8.46	23.32
2 people in an open room	7.73	11.77

5.3 People Detection module

To test the people detection module with the physical robot, the hardware was attached to the MiR100 platform, while keeping it stationary. A person was then put in different locations in front of the cameras to test the reliability of the tracker with both cameras. Here it was found both location and orientation were performing as shown in figure 5.8. It can be seen that the position is staying within 0.05m and the orientation within 0.36rad. The shift in value happening in the middle of figure 5.8 are caused by the test subject turning and thereby changing the physical location of the points utilised for calculation. This is due to the setup of the detector and is to be expected for movement.

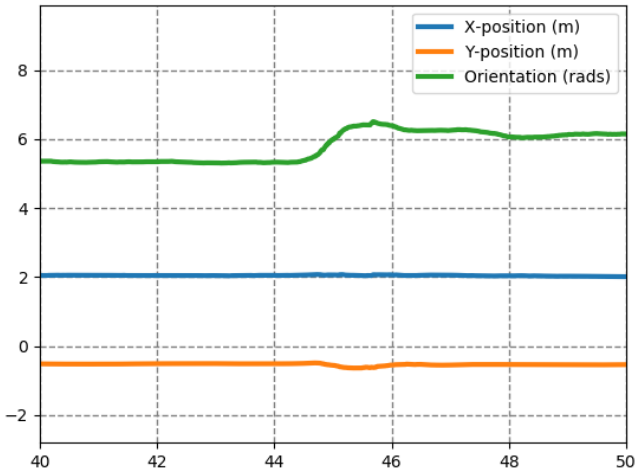


Figure 5.8: Detection of a person turning on the spot for reliability of the Multi Person Tracker. Here the blue graph is the X-position in meters, orange is Y position in meters and green is the orientation in radians. The x-axis represents time in seconds.

To ensure that the detector is able to detect people within the distances, specified in section 2.5.1, two tests were performed. In the first test, a test subject was standing in front of the cameras closer than 1m and slowly walking backwards to determine when the person was detected. From the graph in figure 5.9 it can be seen that the detection is happening at a distance of 1.08m. From the graph it can also be seen that the person is lost as soon as they get within 1.08m, which results in the effective range of detection starting at that distance.

Similarly, the second test was performed to determine if the People Detector can detect a person at 5m or further away. Here the person started walking into the FOV of the RealSense cameras and walked further away before turning and walking towards the robot. In figure 5.10 it can be seen that the person is detected further away than 5.5m and keeps being detected while turning and walking back towards the robot.

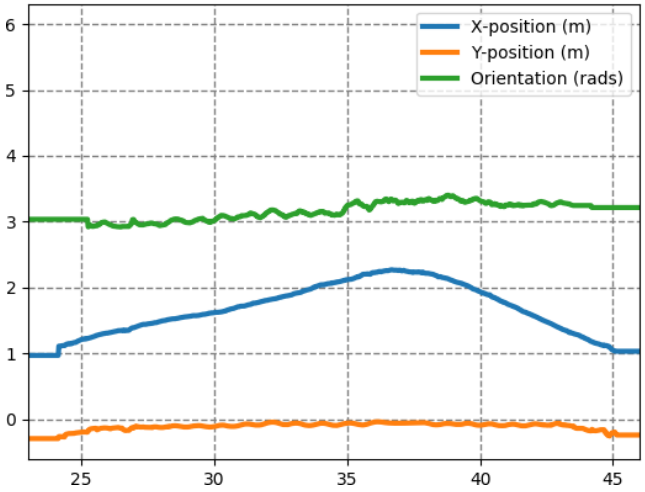


Figure 5.9: The performance of the People Detection module when a person is moving within 1m of the robot. Here the blue graph is the X-position in meters, orange is Y position in meters and green is the orientation in radians. The x-axis represents time in seconds.

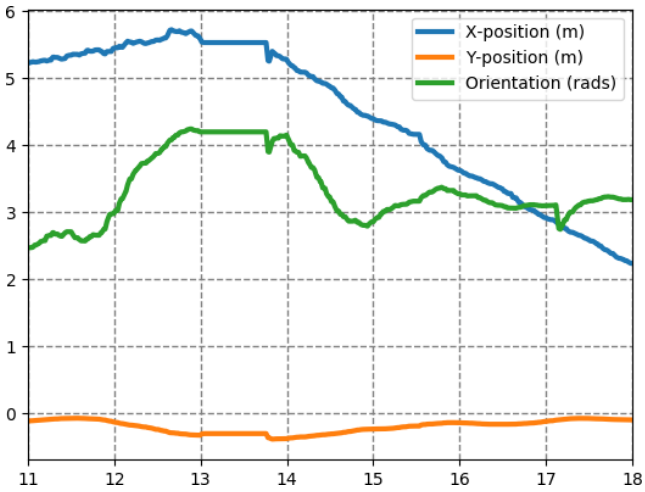
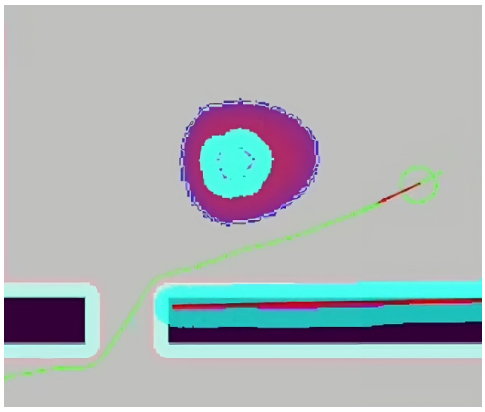


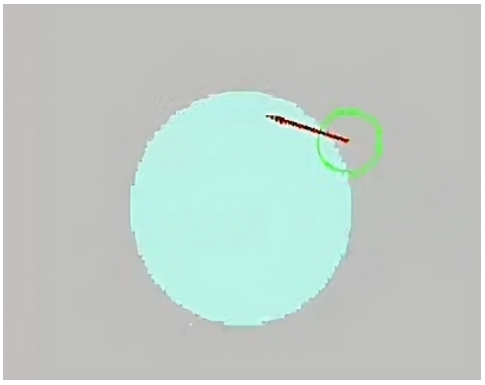
Figure 5.10: The performance of the People Detection Module when a person is further than 5m away and moving towards the robot. Here the blue graph is the X-position in meters, orange is Y position in meters and green is the orientation in radians. The x-axis represents time in seconds.

5.3.1 Using people detection module with simulation

In addition to testing the people detection module on the stationary MiR100 platform, a test to see how the people detection worked in collaboration with the rest of Nav2CAN. This is done by using the RealSense cameras on a stationary setup and running the simulation environment in Stage. Using this with a static tf_transform between *map* and *camera_link* allows to publish the people, who are standing in front of the camera, into the simulation. This allows to show a proof of concept of Nav2CAN where both the *social_layer* and the *interaction_layer* can be used for navigation. Figure 5.11 show examples of the costmaps from the simulation. The figure shows how the robot is able to navigate around the people in the map.



(a) A person detected from the RealSense published into the simulation for the robot to navigate around.



(b) An interaction zone published into the simulation with the robot staying at the border for the interaction.

Figure 5.11: Examples of the costmap layers from a a person detected in the real world, used in the navigation.

5.4 Nav2CAN on MiR100

As described in section 4.6 bridging between ROS1 and ROS2 and potentially a robot can add a lot of complexity to a system. In the pursuit to perform bridging between the MiR100 platform and the Nav2 stack with Nav2CAN it was found that some topics had to be modified to perform correctly between the two. This mainly applies to "latched" topics since the *ros1_bridge* has a bug that does not republish said topics using the correct Quality of Service (QoS) profile on the ROS2 side, resulting in the map onlly appearing once. This issue surfaced e.g. in the case of the *map* topic. The solution to this is to run a ROS2 node before starting the *ros1_bridge* that listens to the map topic and republishes the map using a QoS

profile using transient-local durability and keep-all history.

This allowed to use the platform for default navigation, without any context awareness. However, when introducing the detector module from section 4.2 as well as the context module from section 4.3 more issues surface. These are *tf_transformation* issues where the other nodes do not have sufficient accessibility to the transforms published by the platform, which might be caused by networking issues. Therefore the location of the detected people cannot be transformed into the *map* or *odom* frame rendering the functionality useless.

Due to time constraints and the sheer amount of connections that had to be bridged, Nav2CAN was unable to be tested onboard the system during this project. It was found during the bridging of simulation that the robot controller should only publish sensor data and subscribe to velocity commands while the rest of navigation should be performed on the Nav2 stack.

As the MiR platform is closed off from changing parameters within the system, the possibilities are limited to not bridging topics between ROS1 and ROS2 and ensuring that every necessary topic, such as transformations, sensor fusion, and a map, is run within ROS2. Running ROS2 nodes for all these systems is time consuming to implement and was therefore not performed during this project. Instead, the authors of this project would recommend to use robotic platforms that support ROS2 when developing a project for Nav2 or ensuring that the bridging is possible in a similar manner as done in the simulation, described in section 4.5.

6 - Discussion and conclusion

In this chapter the discussion and conclusion of this project is found. Here the different methods used and compared are discussed for potential benefits and disadvantages of using Nav2CAN. Furthermore, this chapter contains the conclusion of the project where the final system is compared to the requirements of the project.

6.1 Discussion

The discussion seeks to examine the results of the different implemented system and how it performs in the described scenarios. Furthermore, the improvements that can be made to the system along with adaptions to make it perform better in certain environments and contexts are addressed.

As described in chapters 1 and 2, context-awareness in mobile robotics is described as the ability to understand and respond to a given scenario in the operational environment. This includes the static environment, dynamic obstacles, and humans. Understanding and reacting to humans in the environment means that the robot is programmed to adhere to social norms where the actions of the robot can be anticipated by the users. Additionally, the robot must be able to adapt to the actions of the surrounding users. This is desirable to ensure that robot is utilised instead of being seen as an uncomfortable tool by the users.

Many approaches to context awareness have been described, each with different focuses that are often limited to a given type of environment. Nav2CAN seeks to achieve a more comfortable navigation of mobile platforms by understanding social zones along with HHI to achieve non-disturbing navigation. By implementing additions to the Nav2 stack, used in ROS2, Nav2CAN seeks to enable compatibility with a wider range of planners.

6.1.1 Modularity

As described in chapter 4, Nav2CAN is implemented in a modular setup, meaning that different modules can be used independently as long as the requirements for the modules are met. This means that the system is developed for detection of social zones using any type of sensor as long as the resulting data represents the human in the map using location and orientation. This also means that any

transformations between robot and map must be reliable enough for the robot to locate the detected people accordingly.

The Context-Aware module, described in section 4.3, is developed for F-formation interactions with a limited amount of humans. However, the setup can be used to implement additional types of interaction by re-training the interaction detector with additional data, potentially adding new classes for different interactions. In doing so new shapes of interaction zones can be implemented to suit a given need for implementation.

6.1.2 Timing

An important part of having a navigation system performing reliably is having it running at a high enough rate to achieve real time navigation. As described by the authors of CoHAN [75], an update rate of the navigation stack of 10hz results in real time operation. This aligns with the requirements of this project for update rate, described in section 2.5.1.

As described in section 5.1 the combined runtime of all developed modules in Nav2CAN accumulates to 88.66ms which results in an update rate of more than 10hz. Accounting for overhead in the ROS communication, this results in a sufficiently fast system. It is important to emphasise that the modules are running asynchronously and some modules may perform multiple iterations within the 88.66ms. This means that the social map generator may produce multiple maps before an interactions has been detected. This is by design to ensure that the social zones are update frequently, leading to a safer and more comfortable navigation while achieving interaction detection at a sufficient rate.

The Interaction detector is one of the slowest modules due to the fact that it is run in PyTorch [107] and therefore not as optimised as the detector in the Multi Person Tracker. This can be optimised by converting to TensorRT [95] which is optimised for use on Nvidia hardware. This could results in a significantly higher performance rate.

As explained in section 4.3.1, the Social Map Generator generates images, however it is running on the CPU of the controller and is implemented as a ROS2 python node, which both adds computational overhead. Rewriting this node in C++ and/or GPU accelerated execution could result in significant improvements of the computation time.

As the timings are performed in a combination of the modules detecting between none and two people, the averages might depend on this amount of people in the scene. It would therefore important in the future to know any limitations for the modules when multiple people are present to the scene.

To achieve a high computation rate, the hardware used in this project enables GPU acceleration which results in faster computation for parallel processes. This

enables the computation to be done on the edge instead of in the cloud. This however, means that Nav2CAN requires certain hardware to achieve the presented runtime.

6.1.3 Comparison

As more and more mobile robots become available, more potential interactions and encounters between people and robots may occur. Therefore, it is important that the design of navigation is considered heavily when implementing such robots. As described in chapter 5, CoHAN and Nav2CAN use different approaches to navigate in human filled environments. Different types of comparisons are carried out to determine any similarities and differences between the systems.

It can be noticed in the different scenarios that the social zones of the two systems vary, which is a result of design for implementing the systems. For CoHAN, the design favours a 1m distance to the front of the person while trying to limit the distance to the back of a person within 1m with lower cost. For Nav2CAN, the implementation of social zones adheres to the zones described in section 2.1.1, meaning that more space is preferred in front of a person. This leads to different behaviour when movement is not possible behind a person. As can be seen in section 5.2.2, when two people are standing close in personal distance to each other, both systems plan a path behind one person, keeping different distances with CoHAN moving closer. When there is no longer room behind people or the space is limiting the movement, CoHAN will still try to plan its path close to the humans, even driving straight between people, while Nav2CAN will plan a different path if possible.

The results given in table 5.2 shows the minimum distance between human and robot in a given scenario using the two different navigation systems. As these are measured from centre of robot to centre of person, the actual distance between the edge of the robot and the outer of a person can be lowered by 33.4cm, which is the minimum radius of the PR2 robot. This means that CoHAN get as close as 39cm to a person while Nav2CAN gets within 60cm of a person. As CoHAN manages to achieve this distance in front of a person, this can be seen as uncomfortable and could result in people not wanting to use such a system within a closed facility.

It is also worth mentioning that the interaction detection has only been trained on two-person interactions but generalised to four-person detections as shown in section 5.2.2. This also highlights the great benefits of using a CNN for the detection of interactions, as unforeseen situations can still be classified with similar looking formations.

When comparing CoHAN to Nav2CAN it is important to consider the poten-

tial use in the future. As CoHAN is developed using ROS1 which is nearing its end-of-life phase in 2025, it is important to develop for the future. Furthermore, some underlying systems for CoHAN are not maintained which might result in an unusable system in the future. Nav2CAN is developed using the (at time of development) newest version of ROS2. ROS2 has no known end-of-life date, which might result in a usable system for longer with potential for further improvement and added functionality in the future.

6.1.4 Physical setup

As described in section 5.3, the detection of people can be performed at a distance of 1.08m and goes beyond the desired distance of 5m. This closest distance is relevant to ensure a safe navigation around people by detecting a human before the robot gets too close. It is however important to consider the physical location of cameras on the robot as well as their orientation to achieve a satisfying detection. This means that the cameras can be tilted up or down to change when a person is within FOV as well as considering the use of cameras with larger FOV. As described in 4.2.2, the overlap of multiple cameras in front of the robot ensures reliable detection where the robot will move, however tilting one or both RealSense cameras may result in even better detection, closer than 1m.

Based on the problems described in section 5.4 and the time constraints associated with the Nav2CAN project the implementation has not been completed on the MiR100 platform, as workarounds would have required an unreasonable amount of time. Therefore, some interactions were not tested in the closed environment of the laboratory. However, using the simulation as well as testing individual modules has shown promising utilisation of Nav2CAN as a plugin for Nav2.

6.1.5 Future work

As Nav2CAN was developed during one semester, many different additions and improvements can be made to further increase the functionality of the system. Some of these potential improvements are described briefly in the following:

Detection all around the robot As the people detection module, described in section 4.2, is developed for using RealSense cameras, the system is limited to the amount of cameras connected to the robot. This means that one must acquire enough cameras to cover all angles around the robot to use the system for detecting humans. Alternatively, different approaches for detecting human in the environment can be used, either on their own or as an addition to the RealSense cameras. Such systems could be a leg detector that would help detect humans

using the laser scanners on the robot. This would give locations of the people, potentially with a rough estimate of the orientation of these. This however might not be sufficient for detecting interactions as the orientation might be too unreliable.

Another approach could be to use surveillance cameras within the robots environment to provide location and orientation of people. This could be used to provide the location and orientation of all people at once with the robot applying those within planning range. This would mean that a global path could adhere to more humans or achieve fewer interruptions of human interaction. Additionally, using surveillance has the potential to allow detection of individuals who might otherwise be obstructed from the viewpoint of the robot and would therefore go unnoticed. This approach is similar to the setup described in section 5.3.1. Surveillance further limits the issues of the robot not being able to detect humans that are too close to the robot, as discussed above.

Asking for permission to pass In certain situations, the planner may be unable to provide a path that will not disrupt interaction - this can be in tight corridors or with a larger group of people. In such situations, the robot should be able to ask for permission to move within the interaction zone. Such a system requires the robot to change the interaction zone in such a way that a navigate-able area opens up for the planner after an explicit permission is given from the humans. Such a permission can be recognised through different modalities and should therefore be able to handle speech and gestures along with other potential interactions, such as a pressed button. As Nav2 utilises a behaviour tree this would enable a node to implemented for checking when permission should be granted.

Interaction Variety As mentioned previously, both the *Social Map Generator* and the *Interaction Detection* can be expanded with more possible interactions. This would allow navigation that is aware of not only HHI, but potentially also Human-Object Interaction (HOI) and HRI to be even less disruptive for humans working in the environment. Additionally more information about the humans themselves could be included into the *Social Map Generator* for example their velocities and further proxemics introduced in section 2.1.1. This might make the detection of interactions more robust.

Inclusion of temporal information When detecting interactions of humans in the environment, the inclusion of temporal information might be of great benefit. This would allow to remove false positives when people are e.g. only passing each other. This could help refining the detection process for a more crowded environment where people may move close to each other often without interaction.

6.2 Conclusion

In a previous work [3], the authors of this project have shown that indication of a robot's intended path greatly improves the comfort of humans around the robot. Nav2CAN builds upon the findings of that work by introducing a modular system for awareness of HHI for Nav2, according to the problem formulation defined in chapter 2:

How can a modular system for context aware navigation be developed for edge computation on a mobile robot?.

Interacting with the environment in a context aware manner includes two parts:

- Detection of the environment
- Reacting based on the detections

Nav2CAN introduces an approach to both of these in a modular and extendable fashion, such that they can be repurposed or exchanged based on the application. As Nav2CAN seeks to apply context awareness to the navigation of a mobile platform, more requirements are introduced in section 2.5.1.

In accordance with the requirements, Nav2CAN is able to detect and track the pose of humans in a range of 1.08m up to more than 5m in a field of view of 87.4° in front of the robot. Furthermore Nav2CAN is able to detect interactions based on proxemic zones, leading to an accumulated runtime of 88.66ms.

Nav2CAN is not the first work in the field of human- or social-aware navigation and is therefore compared to the established human aware navigation framework CoHAN Planner [75] in section 5.2 to identify strengths and/or weaknesses of either solution. These comparisons highlight that while CoHAN planner is able to handle static and dynamic people it lacks the awareness of interactions and disrupts these. Nav2CAN on the other hand, is able to detect and act on humans and HHIs in real time as shown in chapter 5.

In summary, Nav2CAN provides a modular and adaptable set of plugins for Nav2 to enable real-time and safe context aware navigation of mobile robots in the presence of humans.

Bibliography

[1] Merriam-Webster.com Dictionary. *"Context."* URL: https://www.merriam-webster.com/dictionary/context (visited on 02/20/2023).

[2] Christoforos I. Mavrogiannis, Francesca Baldini, Allan Wang, Dapeng Zhao, Pete Trautman, Aaron Steinfeld, and Jean Oh. "Core Challenges of Social Robot Navigation: A Survey". In: *CoRR* abs/2103.05668 (2021). arXiv: 2103.05668. URL: https://arxiv.org/abs/2103.05668.

[3] Alexandru M. Smau, Christian W. Rønnest, Jonathan E. Schmidt, and Tristan Schwörer. *MASON - Mobile robot that's Aware of Social Occupancy and Norms*. Semester project report for the 2nd semester of Robotics MSc at Aalborg University. 2022.

[4] Séverin Lemaignan, Mathieu Warnier, E Akin Sisbot, Aurélie Clodic, and Rachid Alami. "Artificial cognition for social human–robot interaction: An implementation". In: *Artificial Intelligence* 247 (2017), pp. 45–69.

[5] Francisco Marques, Duarte Gonçalves, José Barata, and Pedro Santana. "Human-aware navigation for autonomous mobile robots for intra-factory logistics". In: *Symbiotic Interaction: 6th International Workshop, Symbiotic 2017, Eindhoven, The Netherlands, December 18–19, 2017, Revised Selected Papers 6*. Springer. 2018, pp. 79–85.

[6] Jonatan Ginés Clavero, Francisco Martín Rico, Francisco J Rodríguez-Lera, José Miguel Guerrero Hernández, and Vicente Matellán Olivera. "Defining Adaptive Proxemic Zones for Activity-Aware Navigation". In: *Workshop of Physical Agents*. Springer, Cham. 2020, pp. 3–17.

[7] ET Hall. "The hidden dimension. An anthropologist examines man's use of space in public and private. New York: Anchor Books; Doubleday & Company, Inc". In: (1969).

[8] Leslie A Hayduk. "The shape of personal space: An experimental investigation." In: *Canadian Journal of Behavioural Science/Revue canadienne des sciences du comportement* 13.1 (1981), p. 87.

[9] Martin Gérin-Lajoie, Carol L Richards, Joyce Fung, and Bradford J McFadyen. "Characteristics of personal space during obstacle circumvention in physical and virtual environments". In: *Gait & posture* 27.2 (2008), pp. 239–247.

[10] Marie-Lou Barnaud, Nicolas Morgado, Richard Palluel-Germain, Julien Diard, and Anne Spalanzani. "Proxemics models for human-aware navigation in robotics: Grounding interaction and personal space models in experimental data from psychology". In: *Proceedings of the 3rd IROS'2014 workshop "Assistance and Service Robotics in a Human Environment"*. Chicago, United States, 2014. URL: https://hal.science/hal-01082517.

[11] Michael G Efran and James A Cheyne. "Shared space: The co-operative control of spatial areas by two interacting individuals." In: *Canadian Journal of Behavioural Science/Revue canadienne des sciences du comportement* 5.3 (1973), p. 201.

[12] Adam Kendon. "Spacing and orientation in co-present interaction". In: *Development of Multimodal Interfaces: Active Listening and Synchrony: Second COST 2102 International Training School, Dublin, Ireland, March 23-27, 2009, Revised Selected Papers* (2010), pp. 1–15.

[13] Saul Greenberg, Nicolai Marquardt, Till Ballendat, Rob Diaz-Marino, and Miaosen Wang. "Proxemic interactions: the new ubicomp?" In: *interactions* 18.1 (2011), pp. 42–50.

[14] Ioannis Kostavelis and Antonios Gasteratos. "Semantic mapping for mobile robotics tasks: A survey". In: *Robotics and Autonomous Systems* 66 (2015), pp. 86–103.

[15] Klaas Klasing, Georgios Lidoris, Andrea Bauer, Florian Rohrmüller, Dirk Wollherr, and Martin Buss. "The autonomous city explorer: Towards semantic navigation in urban environments". In: *1st Int. Workshop on Cognition for Technical Systems*. 2008.

[16] Sourav Garg, Niko Sünderhauf, Feras Dayoub, Douglas Morrison, Akansel Cosgun, Gustavo Carneiro, Qi Wu, Tat-Jun Chin, Ian Reid, Stephen Gould, et al. "Semantics for robotic mapping, perception and interaction: A survey". In: *Foundations and Trends® in Robotics* 8.1–2 (2020), pp. 1–224.

[17] Dapeng Du, Limin Wang, Huiling Wang, Kai Zhao, and Gangshan Wu. "Translate-to-Recognize Networks for RGB-D Scene Recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[18] Xiaojun Chang, Zhigang Ma, Yi Yang, Zhiqiang Zeng, and Alexander G. Hauptmann. "Bi-Level Semantic Representation Analysis for Multimedia Event Detection". In: *IEEE Transactions on Cybernetics* 47.5 (2017), pp. 1180–1197. DOI: 10.1109/TCYB.2016.2539546.

[19] Vincent S. Chen, Paroma Varma, Ranjay Krishna, Michael Bernstein, Christopher Re, and Li Fei-Fei. "Scene Graph Prediction With Limited Labels". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[20] Jiuxiang Gu, Handong Zhao, Zhe Lin, Sheng Li, Jianfei Cai, and Mingyang Ling. "Scene Graph Generation With External Knowledge and Image Reconstruction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[21] Xiaojun Chang, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alex Hauptmann. "A comprehensive survey of scene graphs: Generation and application". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.1 (2021), pp. 1–26.

[22] Abel Gawel, Carlo Del Don, Roland Siegwart, Juan Nieto, and Cesar Cadena. "X-View: Graph-Based Semantic Multi-View Localization". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1687–1694. DOI: 10.1109/LRA.2018.2801879.

[23] Sourav Garg, Niko Suenderhauf, and Michael Milford. *LoST? Appearance-Invariant Place Recognition for Opposite Viewpoints using Visual Semantics*. 2018. DOI: 10.48550/ARXIV.1804.05526. URL: https://arxiv.org/abs/1804.05526.

[24] Yufeng Yue, Chunyang Zhao, Ruilin Li, Chule Yang, Jun Zhang, Mingxing Wen, Yuanzhe Wang, and Danwei Wang. "A Hierarchical Framework for Collaborative Probabilistic Semantic Mapping". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9659–9665. DOI: 10.1109/ICRA40945.2020.9197261.

[25] Robert Goeddel and Edwin Olson. "Learning semantic place labels from occupancy grids using CNNs". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 3999–4004. DOI: 10.1109/IROS.2016.7759589.

[26] Yiyi Liao, Sarath Kodagoda, Yue Wang, Lei Shi, and Yong Liu. "Understand scene categories by objects: A semantic regularized scene classifier using Convolutional Neural Networks". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 2318–2325. DOI: 10.1109/ICRA.2016.7487381.

[27] Massimiliano Mancini, Samuel Rota Bulò, Barbara Caputo, and Elisa Ricci. "Robust Place Categorization With Deep Domain Generalization". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2093–2100. DOI: 10.1109/LRA.2018.2809700.

[28] Kaiyu Zheng and Andrzej Pronobis. "From Pixels to Buildings: End-to-end Probabilistic Deep Networks for Large-scale Semantic Mapping". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 3511–3518. DOI: 10.1109/IROS40897.2019.8967568.

[29] Margarita Grinvald, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Cesar Cadena, Roland Siegwart, and Juan Nieto. "Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery". In: *IEEE Robotics and Automation Letters* 4.3 (2019), pp. 3037–3044. DOI: 10.1109/LRA.2019.2923960.

[30] Ue-Hwan Kim, Jin-Man Park, Taek-jin Song, and Jong-Hwan Kim. "3-D Scene Graph: A Sparse and Semantic Representation of Physical Environments for Intelligent Agents". In: *IEEE Transactions on Cybernetics* 50.12 (2020), pp. 4921–4933. DOI: 10.1109/TCYB.2019.2931042.

[31] İlker Bozcan and Sinan Kalkan. "COSMO: Contextualized scene modeling with Boltzmann Machines". In: *Robotics and Autonomous Systems* 113 (2019), pp. 132–148. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2018.12.009. URL: https://www.sciencedirect.com/science/article/pii/S0921889018303427.

[32] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. "3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans". In: *arXiv preprint arXiv:2002.06289* (2020).

[33] Joao Carreira and Andrew Zisserman. "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[34] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. "A short note about kinetics-600". In: *arXiv:1808.01340* (2018).

[35] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. "A short note on the kinetics-700 human action dataset". In: *arXiv:1907.06987* (2019).

[36] Lucas Smaira, João Carreira, Eric Noland, Ellen Clancy, Amy Wu, and Andrew Zisserman. "A short note on the kinetics-700-2020 human action dataset". In: *arXiv preprint arXiv:2010.10864* (2020).

[37] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. "Learning task-oriented grasping for tool manipulation from simulated self-supervision". In: *The International Journal of Robotics Research* 39.2-3 (2020), pp. 202–216.

[38] Mehmet R. Dogar, Maya Cakmak, Emre Ugur, and Erol Sahin. "From primitive behaviors to goal-directed behavior using affordances". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, pp. 729–734. DOI: 10.1109/IROS.2007.4399469.

[39] James J Gibson. *The ecological approach to visual perception: classic edition*. Psychology press, 2014.

[40] Stefanie Tellex, Ross Knepper, Adrian Li, Daniela Rus, and Nicholas Roy. "Asking for help using inverse semantics". In: (2014).

[41] Ze Gong and Yu Zhang. "Temporal Spatial Inverse Semantics for Robots Communicating with Humans". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 4451–4458. DOI: 10.1109/ICRA.2018.8460754.

[42] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. "Visual place recognition: A survey". In: *ieee transactions on robotics* 32.1 (2015), pp. 1–19.

[43] Nina Savela, Tuuli Turja, and Atte Oksanen. "Social acceptance of robots in different occupational fields: a systematic literature review". In: *International Journal of Social Robotics* 10.4 (2018), pp. 493–502.

[44] Shoko Fuji, Mai Date, Yuko Nagai, Yuko Yasuhara, Tetsuya Tanioka, and Fuji Ren. "Research on the possibility of humanoid robots to assist in medical activities in nursing homes and convalescent wards". In: *2011 7th International Conference on Natural Language Processing and Knowledge Engineering*. 2011, pp. 459–463. DOI: 10.1109/NLPKE.2011.6138243.

[45] Theresa Law, Josh de Leeuw, and John H Long. "How movements of a non-humanoid robot affect emotional perceptions and trust". In: *International Journal of Social Robotics* 13 (2021), pp. 1967–1978.

[46] Christoforos Mavrogiannis, Patrícia Alves-Oliveira, Wil Thomason, and Ross A Knepper. "Social momentum: Design and evaluation of a framework for socially competent robot navigation". In: *ACM Transactions on Human-Robot Interaction (THRI)* 11.2 (2022), pp. 1–37.

[47] Franziska Babel, Andrea Vogt, Philipp Hock, Johannes Kraus, Florian Angerer, Tina Seufert, and Martin Baumann. "Step aside! VR-based evaluation of adaptive robot conflict resolution strategies for domestic service robots". In: *International Journal of Social Robotics* 14.5 (2022), pp. 1239–1260.

[48] Franziska Babel, Johannes M. Kraus, and Martin Baumann. "Development and Testing of Psychological Conflict Resolution Strategies for Assertive Robots to Resolve Human–Robot Goal Conflict". In: *Frontiers in Robotics and AI* 7 (2021). ISSN: 2296-9144. DOI: 10.3389/frobt.2020.591448. URL: https://www.frontiersin.org/articles/10.3389/frobt.2020.591448.

[49] Tatsuya Nomura and Kazuma Saeki. "Effects of polite behaviors expressed by robots: A psychological experiment in Japan". In: *International Journal of Synthetic Emotions (IJSE)* 1.2 (2010), pp. 38–52.

[50] Aimi Shazwani Ghazali, Jaap Ham, Emilia Barakova, and Panos Markopoulos. "Persuasive robots acceptance model (PRAM): roles of social responses within the acceptance model of persuasive robots". In: *International Journal of Social Robotics* 12 (2020), pp. 1075–1092.

[51] Annemarie Turnwald and Dirk Wollherr. "Human-like motion planning based on game theoretic decision making". In: *International Journal of Social Robotics* 11 (2019), pp. 151–170.

[52] Pier Luigi Dovesi, Matteo Poggi, Lorenzo Andraghetti, Miquel Martí, Hedvig Kjellström, Alessandro Pieropan, and Stefano Mattoccia. "Real-time semantic stereo matching". In: *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2020, pp. 10780–10787.

[53] Kelvin Wong, Shenlong Wang, Mengye Ren, Ming Liang, and Raquel Urtasun. "Identifying unknown instances for autonomous driving". In: *Conference on Robot Learning*. PMLR. 2020, pp. 384–393.

[54] Ioannis Kostavelis, Manolis Vasileiadis, Evangelos Skartados, Andreas Kargakos, Dimitrios Giakoumis, Christos-Savvas Bouganis, and Dimitrios Tzovaras. "Understanding of human behavior with a robotic agent through daily activity analysis". In: *International Journal of Social Robotics* 11 (2019), pp. 437–462.

[55] Hokyoung Ryu and Andrew Monk. "Interaction unit analysis: A new interaction design framework". In: *Human–Computer Interaction* 24.4 (2009), pp. 367–407.

[56] Angelique Taylor, Darren M Chan, and Laurel D Riek. "Robot-centric perception of human groups". In: *ACM Transactions on Human-Robot Interaction (THRI)* 9.3 (2020), pp. 1–21.

[57] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. "Sophie: An attentive gan for predicting paths compliant to social and physical constraints". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 1349–1358.

[58] Anirudh Vemula, Katharina Muelling, and Jean Oh. "Social attention: Modeling attention in human crowds". In: *2018 IEEE international Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 4601–4607.

[59] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. "Visual social relationship recognition". In: *International Journal of Computer Vision* 128 (2020), pp. 1750–1764.

[60] Meng Zhang, Xinchen Liu, Wu Liu, Anfu Zhou, Huadong Ma, and Tao Mei. "Multi-granularity reasoning for social relation recognition from images". In: *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2019, pp. 1618–1623.

[61] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. "Action genome: Actions as compositions of spatio-temporal scene graphs". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10236–10247.

[62] Yi-Fan Song, Zhang Zhang, Caifeng Shan, and Liang Wang. "Richly activated graph convolutional network for robust skeleton-based action recognition". In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.5 (2020), pp. 1915–1925.

[63] Karim Youssef, Sherif Said, Samer Alkork, and Taha Beyrouthy. "A Survey on Recent Advances in Social Robotics". In: *Robotics* 11.4 (2022), p. 75.

[64] Rico Francisco Martiín. *A concise introduction to robot programming in ROS2*. CRC Press, 2023.

[65] Open Robotics. *ROS2 Documentation: Humble*. URL: https://docs.ros.org/en/humble (visited on 04/20/2023).

[66] Open Robotics. *ROS Noetic Ninjemys*. URL: http://wiki.ros.org/noetic (visited on 04/20/2023).

[67] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. "Robot Operating System 2: Design, architecture, and uses in the wild". In: *Science Robotics* 7.66 (2022). DOI: 10.1126/scirobotics.abm6074. URL: https://doi.org/10.1126\%2Fscirobotics.abm6074.

[68] Tully Foote. "tf: The transform library". In: *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*. Open-Source Software workshop. 2013, pp. 1–6. DOI: 10.1109/TePRA.2013.6556373.

[69] Steven Macenski, Francisco Martin, Ruffin White, and Jonatan Ginés Clavero. "The Marathon 2: A Navigation System". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[70] Michael Ferguson. *navigation*. URL: http://wiki.ros.org/navigation (visited on 04/20/2023).

[71] ros planning. *Tuning*. URL: https://navigation.ros.org/tuning/index.html (visited on 05/26/2023).

[72] Michael Ferguson. *RViz*. URL: http://wiki.ros.org/rviz (visited on 04/20/2023).

[73] Michael Ferguson. *PlotJuggler*. URL: http://wiki.ros.org/plotjuggler (visited on 04/20/2023).

[74] Michael Ferguson. *rqt*. URL: http://wiki.ros.org/rqt (visited on 04/20/2023).

[75] Phani Teja Singamaneni, Anthony Favier, and Rachid Alami. "Human-Aware Navigation Planner for Diverse Human-Robot Ineraction Contexts". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021.

[76] Phani Teja Singamaneni and Rachid Alami. "HATEB-2: Reactive Planning and Decision making in Human-Robot Co-navigation". In: *International Conference on Robot & Human Interactive Communication*. 2020. DOI: 10.1109/RO-MAN47096.2020.9223463.

[77] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 269–286.

[78] Xin Li, Kejun Wang, Wei Wang, and Yang Li. "A multiple object tracking method using Kalman filter". In: *The 2010 IEEE international conference on information and automation*. IEEE. 2010, pp. 1862–1866.

[79] Amaury Auguste, Wissam Kaddah, Marwa Elbouz, Ghislain Oudinet, and Ayman Alfalou. "Behavioral Analysis and Individual Tracking Based on Kalman Filter: Application in an Urban Environment". In: *Sensors* 21.21 (2021). ISSN: 1424-8220. DOI: 10.3390/s21217234. URL: https://www.mdpi.com/1424-8220/21/21/7234.

[80] James Munkres. "Algorithms for the assignment and transportation problems". In: *Journal of the society for industrial and applied mathematics* 5.1 (1957), pp. 32–38.

[81] Algorithm Workshop. *Munkres' Assignment Algorithm*. 2022. URL: https://brc2.com/the-algorithm-workshop/.

[82] H. Choset, Kevin M Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2005.

[83] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. 2022. DOI: 10.48550/ARXIV.2207.02696. URL: https://arxiv.org/abs/2207.02696.

[84] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection". In: *arXiv:2004.10934* (2020).

[85] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[86] *Nvidia Jetson Orin*. URL: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/.

[87] Nvidia. *Nvidia embedded systems for next-Gen Autonomous Machines*. URL: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/.

[88] Intel. *Intel RealSense D400 Series Product Family*. https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf. Accessed on 04.04.2023. 2019.

[89] MiR Mobile industrial Robots. *MiR100*. https://www.mobile-industrial-robots.com/solutions/robots/mir100/. Accessed on 04.04.2023. 2022.

[90] *Safety Laser Scanners S300 standard*. URL: https://www.sick.com/us/en/safety-laser-scanners/safety-laser-scanners/s300-standard/c/g187239.

[91] Universal Robots. *UR5 collaborative robot arm: Flexible and Lightweight Cobot*. URL: https://www.universal-robots.com/products/ur5-robot/.

[92] Intel. *realsense-ros*. URL: https://github.com/IntelRealSense/realsense-ros (visited on 05/22/2023).

[93] Dusty-Nv. *Jetson-inference/posenet.md*. 2023. URL: https://github.com/dusty-nv/jetson-inference/blob/master/docs/posenet.md.

[94] Dusty Nv. *Jetson-inference*. URL: https://github.com/dusty-nv/jetson-inference.

[95] Nvidia. *Nvidia TENSORRT*. 2023. URL: https://docs.nvidia.com/deeplearning/tensorrt/.

[96] Rudolph Emil Kalman. "A new approach to linear filtering and prediction problems". In: (1960).

[97] Jonathan M Blackledge. *Digital signal processing: mathematical and computational methods, software development and applications*. Elsevier, 2006.

[98] Dan Lazewatsky and David V. Lu. *ros people_msgs*. 2021. URL: https://github.com/wg-perception/people/tree/melodic/people_msgs.

[99] Jonatan Gines Clavero, Francisco Martín Rico, Francisco J Rodríguez-Lera, José Miguel Guerrero Hernández, and Vicente Matellán Olivera. "Impact of decision-making system in social navigation". In: *Multimedia Tools and Applications* 81.3 (2022), pp. 3459–3481.

[100] Steven Macenski. *nav2_bringup*. URL: https://github.com/ros-planning/navigation2/tree/main/nav2_bringup (visited on 05/22/2023).

[101] ROS community. *pr2*. URL: http://wiki.ros.org/Robots/PR2 (visited on 05/22/2023).

[102] Open Robotics. *What is morse?* Accessed on 04.04.2023. 2010. URL: https://www.openrobots.org/morse/doc/latest/what_is_morse.html.

[103] Richard Vaughan. *Stage*. 2011. URL: https://rtv.github.io/Stage/.

[104] ros2. *Ros2/ros1_bridge: Ros 2 package that provides bidirectional communication between ROS 1 and Ros 2*. URL: https://github.com/ros2/ros1_bridge.

[105] Martin Günther and contributors. *mir_driver*. URL: https://github.com/DFKI-NI/mir_robot (visited on 05/25/2023).

[106] Sphanit. *Sphanit/cohan_navigation*. 2021. URL: https://github.com/sphanit/CoHAN_Navigation.

[107] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG].