UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

MASTER'S DEGREE IN
COMPUTER ENGINEERING

Shared Control Navigation of TIAGo in a Partially Observable Environment using Artificial Potential Fields and User Intent Prediction

Supervisor**: Dr. Stefano Tortora**
Candidate**: Paria Haji Abolfath**

Academic Year: 2023/2024

Date of Graduation: July 8th, 2024

Abstract

In recent years, the use of autonomous mobile robots with the goal of increasing the productivity in different domains, has been on the rise. The field of AI robotics has seen many advances since its emergence with the early robots that were built in an effort to facilitate some work for humans.

The human-robot interaction can be categorized into three methods of shared-control, shared-autonomy and shared-intelligence. Starting from shared-control and moving towards shared-intelligence, the robot progressively acquires more participation in its own final action. Shared-control, which is also the focus of this thesis work, involves the robot being able to apply small modifications to the user inputs such as altering the steering angle due to the existence of an obstacle. In shared-autonomoy, there exist pre-defined behavioral procedures for the robot that get activated in very certain occasions and moreover, there is some a priori knowledge available about the environment which is provided to the robot. Shared-intelligence can be considered as the method in which the robot has the most *active* role in determining the final decision. It makes use of several policies which are essentially behavioral guidelines for the robot, each policy generates a probability grid which will be fused with every other policy. The final action of the robot will be based on this fusion, which is basically the action, that according to the policies, is considered as the most probable for that situation.

Therefore, the work presented in this thesis explores the topic of shared control navigation of a mobile robot, namely TIAGo from PAL Robotics [1], in a partially observable environment, while the navigation is locally planned using artificial potential fields (APF) created around the objects and the system is assisted by predicting the most probable intention of the user.

The experiments were carried out both in a simulated environment and on the real robot. The behavioral results of the robot were almost similar in the two environments although we did not manage to dive deep enough in the real-world experiments.

The APF showed promising results while the utilized assistance system was originally tested in an object manipulation context and still needs more fine-tuning

for our case of navigation and this would thus be among the future work on this thesis along with providing more in-depth results from the real-world experiments.

Sommario

Negli ultimi anni è aumentato l’uso di *autonomous mobile robots* con l’obiettivo di aumentare la produttività in diversi settori. Il campo della robotica basata sull’intelligenza artificiale ha visto molti progressi sin dalla sua comparsa con i primi robot costruiti nel tentativo di facilitare alcuni lavori per gli esseri umani.
L’interazione uomo-robot può essere classificata in tre metodi di shared-control, shared-autonomy e shared-intelligence. Partendo dal shared-control e procedendo verso il shared-intelligence, il robot acquisisce progressivamente una maggiore partecipazione alla propria azione finale. Il shared-control, che è anche il fulcro di questo lavoro di tesi, prevede che il robot sia in grado di applicare piccole modifiche agli input dell'utente, come alterare l'angolo di sterzata a causa dell'esistenza di un ostacolo. Nel shared-autonomy, esistono procedure comportamentali predefinite per il robot che vengono attivate in determinate occasioni e inoltre, è disponibile una certa conoscenza a priori sull'ambiente che viene fornita al robot. Il shared-intelligence può essere considerato come il metodo in cui il robot ha il ruolo più attivo nel determinare la decisione finale. Fa uso di diverse politiche che sono essenzialmente linee guida comportamentali per il robot, ciascuna politica genera una griglia di probabilità che verrà fusa con ogni altra politica. Su questa fusione si baserà l'azione finale del robot, che è sostanzialmente l'azione che, secondo le policy, è considerata la più probabile per quella situazione.
Pertanto, il lavoro presentato in questa tesi esplora il tema della navigazione a shared-control di un mobile robot, vale a dire TIAGo di PAL Robotics [1], in un ambiente parzialmente osservabile, mentre la navigazione è pianificata localmente utilizzando artificial potential fields (APF) creati attorno agli oggetti e il sistema è assistito prevedendo l'intenzione più probabile dell'utente.
Gli esperimenti sono stati condotti sia in un ambiente simulato che sul robot reale. I risultati comportamentali del robot erano quasi simili nei due ambienti, anche se non siamo riusciti ad approfondire abbastanza gli esperimenti nel mondo reale.
L'APF ha mostrato risultati promettenti mentre il sistema di assistenza utilizzato è stato originariamente testato in un contesto di object manipulation e necessita ancora di una maggiore messa a punto per il nostro caso di navigazione e questo rientrerebbe quindi tra i lavori futuri su questa tesi insieme a fornire risultati più approfonditi dagli esperimenti nel mondo reale.

Contents

Acknowledgements

This thesis was developed at University of Padua, Intelligent Autonomous Systems (IAS) Laboratory between October 2023 and June 2024. The work on the thesis started off as a research training period in the aformentioned laboratory, where I got to initially study the previous works on Artificial Potential Fields and Shared-Control/Autonomy/Intelligence Methods, carried out previously by the members of the laboratory. I wish to thank my supervisor, Prof. Stefano Tortora, for giving me supervision during this time and the IAS Laboratory for providing the access to the TIAGo robot for carrying out the experiments, leading to the final preparation of this thesis.

1. Introduction

1.1 A brief history of AI robotics

In artificial intelligence, an agent, situated in an external world, is an entity sensing its surroundings and taking actions that change the environment as opposed to solely observing and modifying its own internal state. This agent can be situated either in a virtual world (defined by the World Wide Web, a simulation and such) or in a physical one. The former is called a software agent and the latter, a robot.

An intelligent agent, can be defined as a system that perceives its surroundings and takes actions that maximize its chances of success.

Taking into account the definitions provided above, we may define an *intelligent robot* as a physically situated intelligent agent.

The rise of the AI apporach to robotics can be traced to the space race era. In 1961, after the success of the Soviet Union in the Sputnik space program, the United States declared that they would put a man on the moon by 1970. Realizing the potential risks regarding sending humans to the moon such as the hostile environment of the outer space, bulky spacesuits making it difficult for the astronauts to complete their tasks and the dangers of utilizing experimental spacecrafts to carry human beings to the space, paved the way for replacing the astronaut with a planetary rover. [2]

Fig. 1. John Young uncomfortably collecting lunar samples [3]

Due to the limitations of the rover technology such as inability to safely control the rover over the notoriously poor radio links of the time, one possible solution was to have autonomous mobile robots land on a planet, conduct the explorations, complete the requiered tasks and radio back the results to Earth.

These autonomous mobile robots would ideally have a high degree of autonomy. In addition to the robot receiving commands from the Earth and sending back signals, it would also navigate properly and not fall into canyons.
It was soon realized that having a rover working on a planet on its own, would lead to the occurrence of many unpredictable performance issues. The autonomous rover could easily end up finding itself in situations from which it could not save itself.
Therefore, at first robots might have seemed like a temporary solution for putting human beings into space but they became more complicated soon after. NASA introduced the idea that AI robots were much more than simply being bolted to the factory ground for indusrial purposes and that they might have the potential of integrating all forms of AI (understanding speech, planning, reasoning, representing the world, learning) into one program. A program, which of course, has not been made yet. [2]
The field of AI robotics has surely experienced significant advances since then. The field has seen the emergence of the first AI robots such as Shakey; using a “sense-plan-act” cycle, the Sojourner robot; a Mars explorer capable of navigating to a point selected on an image without human intervention and the iRobot Roomba; the vacuum cleaner robot.

Fig. 2. Shakey and Charles Rosen, one of the inventors of the robot [4]

Autonomous mobile robots have gained a dramatic popularity in the recent years, due to their practicality and potential uses in the modern world. They are capable of independent decision-making and taking corrective actions, similar to the behavior

generally seen in humans. To the point that a fully autonomous mobile robot is able to perceive the environment, make judgments based on the received sensory information and/or what is has been trained to recognize and then take an action using all that knowledge.

1.2 Human-robot interaction

It may happen that the human is involved in doing some task that is particularly tiring, boring or generally demanding in time and energy. The human would prefer to delegate those tasks to the robot rather than micromanaging every little motion or decision. Shared approaches, in which the robot receives the user commands and contextualizes them using its context awareness (provided by the sensory information), involves both the human and the robot [5]. The final's robot action is determined by evaluating the shared information among the two, which is essentially the user's commands and the robot's perception.

Three main forms of human-robot interaction has been detected and taxonomized as *shared-control, shared-autonomy* and *shared-intelligence.* The choice of these three typologies recalls the decision-making theory, categorizing the human choices into three levels as follows:

- Operational level: detailed and short term decisions
- Tactical level: including the allocation of resources over a medium planning horizon
- Strategic level: the acquierd strategies for the achievement of high-level goals

Following this classification, we may also deduce a new point of view of the level of details of the decisions taken by the user and by the robot, which is a division into three categories of low level, medium level and high level. Low level interactions are made up of the execution of specific control signals that quickly expire. Medium level interactions refer to performing operations by an autonomous robot in a medium time, while the high level interactions are associated with strategies that aim to guide both the human and the robot's choices in accomplishing their common goal.

It is possible to represent the decision-making theory using a pyramidical structure as observed in figure 4.

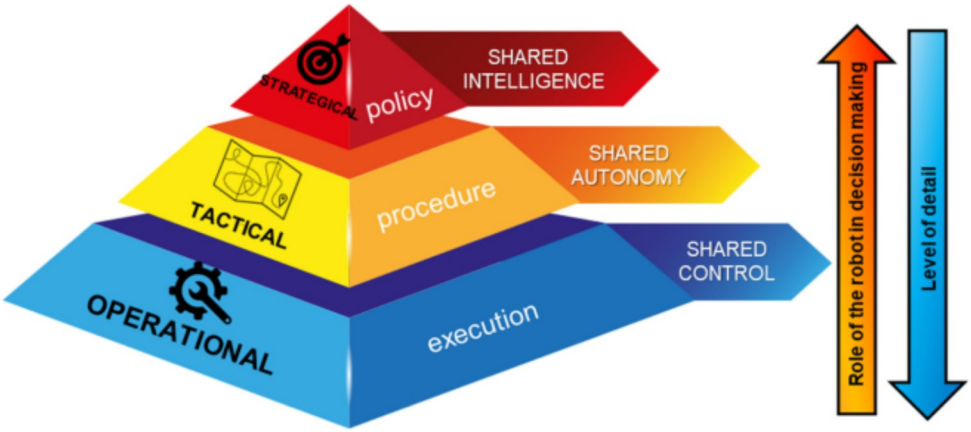

Fig. 3. The decision making theory depicted as a pyramidical structure [5]

As suggested in Figure 4, moving down in the pyramid, we will be dealing with interactions which consist of a less apparent participation of the robot in the decision-making process. While moving up, the robot will be able to make more strategic decisions to guide the shared task with the human. The general structure suggests how the three main approaches of shared-control, shared-autonomy and shared-intelligence differ in terms of the role of the robot in the decision-making process and the level of details of decisions, made by the human and the robot during the whole interaction.
Having the perspective provided in the pyramidical structure above, we may now define the three main shared methods more explicitely.

1.2.1 Shared control

In shared control, both the human and the robot contribute simultaneously to the control. The human provides deliberative inputs and the robot provides reactive control. The user tends to interact at low-level: giving steering/turning commands to the robot. The robot instead, executes those commands by simply applying some adjustments to them for instance modifying the angle of the steering due to the existence of an obstacle nearby. Moreove, in emergency situations the robot stops operating and waits for the user to *save* it from the condition it is stuck in. [5]

1.2.2 Shared autonomy

Shared autonomy is driven by an a prioi knowledge of the environment. The user is still involved in the loop and delivers high-level commands such as setting the final target position of the robot. In other words, it’s the user who takes care of the global

planning phase.The robot receiving these user commands, processes them and treates them as triggers for activating some previously-established routines or procedures.
As opposed to shared control, the user solely supervises the interaction while constantly having the possiblity of intervening in the execution of the shared task.
One vivid difference from the shared control is that since the robot has a map of the environment, it will no longer halt the navigation in case of an emergency. Instead, the robot can activate a recovery procedure to handle the emergency situation.
With the robot being more *actively* environment in the task, the user will only be focusing on the final destination while the robot is managing obstacle avoidance behaviors and trajectory planning.
It is common in shared autonomy methods to make use of a temporary subgoal which keeps getting updated throughout the motion of the robot. This subgoal is defined in an unoccupied cell of the global map. A map that as mentioned before, is provided to the robot and it's the same the robot utilizes for its localization.
Although there is a reduction in the human's workload compared to the pure teleoperation (if not essentially to shared control) and the user can rely more on the capabilities of the robot, shared autonomy has a few shortcomings.
First of all, there is always a setup phase before starting the method which consists of providing the map-related information to the robot. Secondly, the predetermined procedures in the robot are rather limiting. In the sense that there must be a very specific situation in which these behaviors get activated.
Moving higher in the pyramidical structure of Figure 4, we encounter the shared intelligence method which is presented in the next subsection. [5]

1.2.3 Shared Intelligence

In shared intelligence, the robot contributes to the decision making process by questioning the user's commands using its acquired sensory information. The robot is also able to take the control over the human (in emergency cases for instance). Nevertheless, the user is still constantly supervising the task and can intervene at any chosen time.
In these systems, the intelligence of the robot depends on several factors influencing the robot's motion. These factors that are represented as *policies*, each encodes a certain behavioral guideline for the robot such as the robot should stay far from the obstacles, should reach the target and should implement the user's command if possible.
These policies equally contribute in determining the robot's motion. In other words, there is no mechanism choosing strictly one policy in a "winner-takes-it-all" manner.

Each policy generates a probability grid around the robot, these probability grids are fused together and then the result is normalized in order to obtain a final probability grid which covers all local (x,y) positions around the robot. The next subgoal of the robot will be selected from this final probability grid by choosing the cell with the highest probability value. This subgoal will later be given to a navigation module in order to plan the best trajectory for the robot.
This chosen subgoal will be the position towards which the robot will start moving. It's also considered as the most appropriate behavior of the robot in a given scenario.
Therefore, the robot's behavior is not pre-coded as in shared autonomy. Instead, it results from the interactions between the human and the robot working together towards the same goal. [5]
Figure 5 depicts an illustraive scheme of the shared intelligence method.

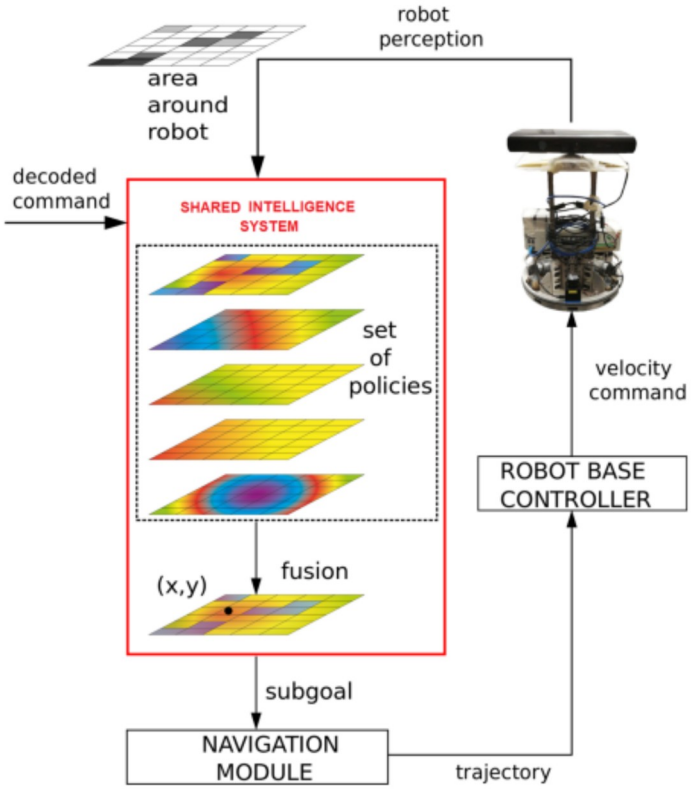

Fig. 4. Typical Shared Intelligence framework [5]

1.3 Autonomous mobile robot navigation

Motion planning of the robot can be defined as the process in which a robot providing services, is require to avoid collisions with obstacles while moving toward a destination. [6]

Motion planning is a broad term and includes different categories of scenarios. A manipulator arm needs motion planning in order to successfully pick a target object and put it in a desired position and orientation, while it should not collide with objects in the scene that are labeled as obstacles. A mobile robot, which as the name suggests, is able to *move around*, needs motion planning as well.
Path planning can be considered as a specific kind of motion planning and is a crucial task in autonomous mobile robots navigation, which requires the robot to find an optimal path based on desired performance outcomes such as shortest time, shortest route and minimum energy consumption. [6]
The path planning problem can be catagorized into global and local path planning. In global path planning, there is assumed to be a complete knowledge about the environment and thus a general path is usually planned offline before the robot starts moving around. In a local path planning scenario instead, it is assumed that the knownledge of the environment is incomplete before the robot starts navigating and therefore, the path is planned in an online manner as the robot roams around and senses the environment. As it can be deduced, local path planners are more useful in situations in which the robot is supposed to navigate in an unknown environment.
In fact, the capability of an autonomous mobile robot equipped with a local planner, is that the robot can adapt its navigation to the changes in the environment and plan paths which are still safe in the sense of not being prone to obstacle collision.
Figure 6 illustrates a condition in which an obstacle has been accidentally put in the middle of the global path planned towards the robot. The robot is able to generate a new local path in order to safely reach the target position.

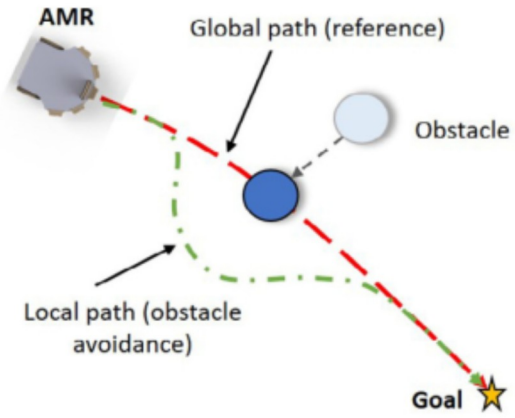

Fig. 5. The robot is able to avoid dynamic obstacles if the path planning is done locally [6]

Path planning techniques for autonomous mobile robots can be divided into classical and heuristic approaches with classical ones being typically more computationally expensive and are usually found less effective in dynamic environments while heuristic approaches are more versitile and can handle uncertainties in the workspace better than the classical methods. The very specific kind of classical path planning that we have focused on in this thesis, is the artificial potential field method which will be well elaborated in the following subsection. [6]

1.4 Artificial potential fields

Artificial potential fields (APF) are widely used for mobile robot path planning. The result of the path planning is usually the direction towards which the robot should turn plus the speed or velocity planning. [7]

The philosophy of the APF approach can be schematically described as follows: “The manipulator (our mobile robot) is moving in a field of forces. The position to be reached is an attractive pole for the end-effector (which in our case is the entire body of the robot) and obstacles are repulsive surfaces for the manipulator parts.” [8]

The idea is to assign a potential value to each point in the robot’s workspace. These potential values represent the influence of difference forces and objects such as obstacles and goals, on the robot. A combination of these forces create a potential field.

Artificial potential fields are a specific kind of potential fields, designed particularly for robotics applications.

In order for us to dive deeper in the context, we might first want to define the problem of collision avoidance with a single obstacle O .

If x_d represents the target position, it is possible to control the robot by subjecting it to the artificial potential field:

$$U_{art}(x) = U_O(x) + U_{xd}(x)$$

Eq. 1.

Leading to the following expression of the potential energy of the Lagrangian:

$$U(x) = U_{art}(x) + U_g(x)$$

Eq. 2.

in which $U_g(x)$ represent the gravity potential energy. Based on Lagrange’s equations and the end-effectory dynamic decoupling [8], we will reach the command vector F (also called the total force F_T) of the decoupled end-effector (in our case, the entire

body of the robot), corresponding to applying the artificial potential field of U_{art} may be written as:

$$F^{*} = F^{*}_{xd} + F^{*}_{O}$$

Eq. 3

where we have:

$$F^{*}_{xd} = -\nabla[U_{xd}(x)]$$
$$F^{*}_{O} = -\nabla[U_{o}(x)]$$

Eq. 4.

F^{*}_{xd} is defined as an attractive force, which allows the point x of the robot, reach the goal position x_d . F^{i}_{O} is the *Force inducing an artificial repulsion from the surface* of the obstacle, which is created by the potential field $U_O(x)$.

The total force (F^{*} or F_T) that is being applied to the robot, is the net force of the attractive as well as repulsive forces. Thanks to this force, the robot is able to steer towards the target position while not colliding with the obstacles.

Ultimately, it will be the gradient of F_T steering the robot in the appropriate direction. Which is, towards the goal and away from the obstacles. We might redefine the attractive force as $F_A(x)$ and the repulsive force as $F_R(x)$:

$$F_T(x) = F_A(x) + F_R(x)$$

Eq. 5.

and thus we will have:

$$F_T(x) = -\nabla[U_{xd}(x)] - \nabla[U_O(x)]$$

Eq. 6.

One note on the sign of the gradients in the formula is that in path planning using the APF, the aim is to guide the robot towards a region in the workspace in which the net force is the minimum. This stems from the reality that in the surroundings of an obstacle, the repulsive forces keep getting larger as we approach the obstacle. While in the case of a target, the attractive forces keep getting smaller as the goal is approached. This phenomenon is observed in Figure 7 where we have the attractive forces around the target in *b* and the repulsive forces around the obstacle in *a:*

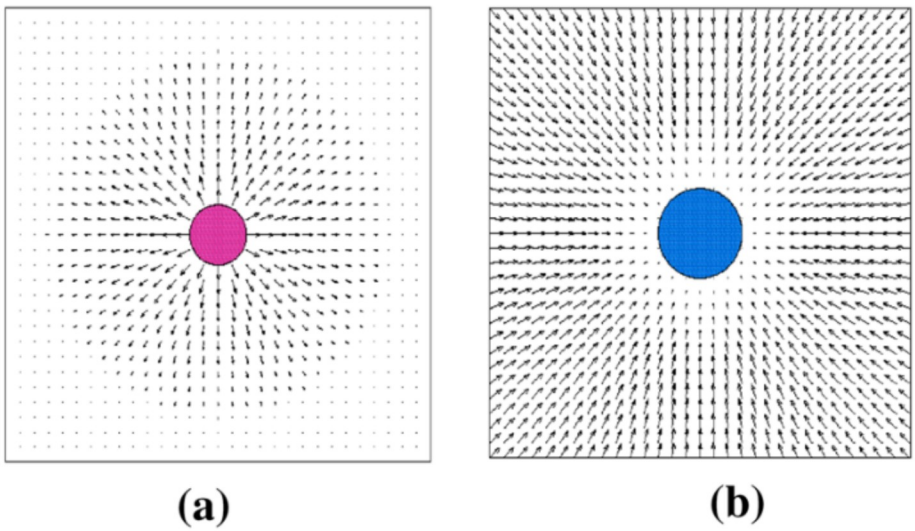

Fig. 6. (a) An obstacle generating a repulsive field in which the energy the energy is maximized at the center of the field. (b) A target object generating an attractive field in which the energy is minimized at the center of the field [9]

in Figure 7 we have the typical setup of a scenario in which there is an encounter between the two types of forces.

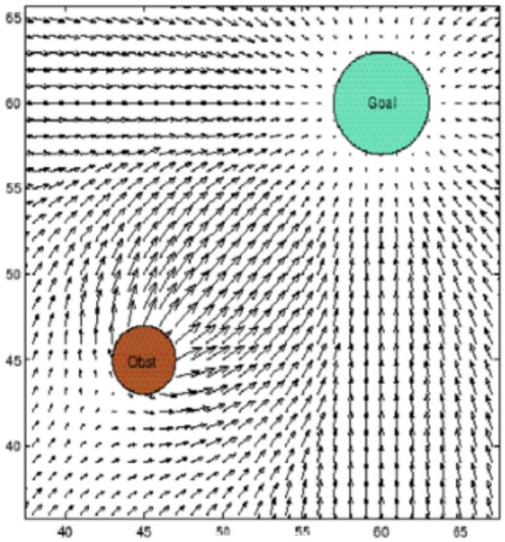

Fig. 7. Interaction between the repulsive and attractive field [9]

As it can be observed in figure 8, the red circle which represents the obstacle is surrounded by repulsive forces (and hence the direction of the tiny arrows which are towards the outside space). While around the green circle which happens to be the goal, there exists a field of attracive forces and therefore, the direction of the arrows are towards the goal itself.
It's worth to note that in the space between the obstacle and the goal where the two fields overlap, there is this pattern which consits of an increase in the repulsion force along with an increase in the attraction force as well. This perfectly models the behavior we would ideally like to see in our path planner, in the vicinity of an obstacle there should be a higher repulsive force to drive the robot further from it while in the vicinity of the goal, we would like have a higher attraction force to guide the robot towards the goal position and not everywhere else.

1.5 Limitations of Artificial Potential Fields

Artificial potential field is no different than any other path planner algorithm in not being the perfect ultimate method, and it surely does suffer from some limitations.
The U_{xd} and U_O are potential functions. There is no doubt that they should be differentiable in order for us to be able to compute the forces the robot will be subjected to. The main issue of artificial potential fields arises from the fact that the method is prone to getting stuck in the local minima of the potential functions.
In figure 9 we see two different repulsive potential fields (both differentiable of course) around the human (obstacle). Having a larger potential (a) increases the safety in terms of obstacle collision while having a smaller potential (b) might increase the efficiency of path planning since there might be a bunch of obstacles gathered together in one region of space and the robot is supposed to move through them.

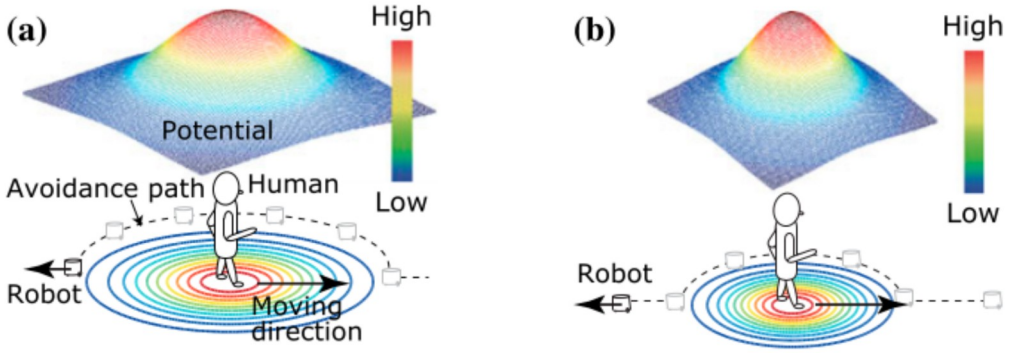

Fig. 8. The choice of the potential function can influence the performance of the system [10]

When computing the derivative of the potential functions, there might be a single global minimum and several local minima where the robot can steer towards.
Behavior-wise, when stuck in a local minimum, the robot has a velocity equal to or almost equal to 0. As seen in Figure 10, in both of the cases the target is attracting the robot along the shortest path possible while the obstacle is repelling it in the opposite direction and therefore, the resulting force becomes almost zero and the robot gets stuck in this condition, not being able to move any further towards or in the opposite direction of the goal.

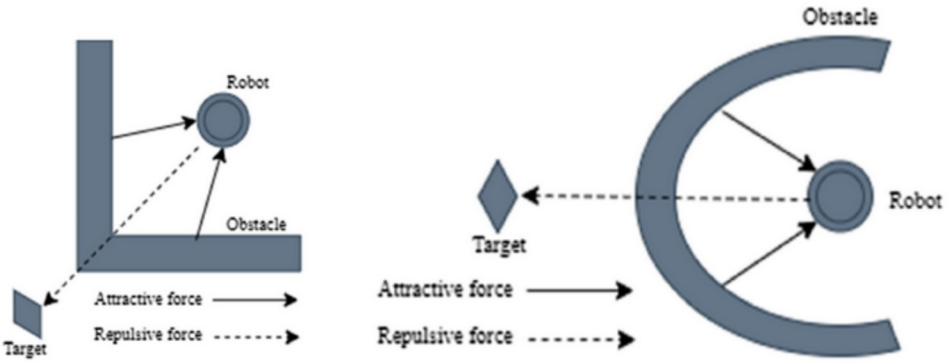

Fig. 9. Two scenarios illustrating how the robot can get trapped inside a local minimum. There exists a target beyond the obstacle and the robot is attracted it [9]

Another issue with the APF is oscillations. This phenomenon occurs when the robot has to navigate itself in a narrow passage. When entering the narrow passage, the robot's sensors perceive the wall on one of the sides of the passage and thus the robot is guided towards the opposite direction of the observed obscale and is therefore steered to the opposite side. Soon after this change in direction, the robot senses the wall on the second side of the passage and repeats the same behavior as for the first wall. This behavioral patten keeps getting repeated and as a result, the robot's motion will not be pretty smooth. It will be oscillating and making small *wobbles* until it gets out of the narrow hallway/passage.
A few methods handling the problem of local minima will be presented in chapter 2.

1.6 Scope of the work

This thesis work starts off by envisioning a scenario in which a mobile robot (TIAGo to be precise) and a user being able to send velocity commands using the keyboard, are working together with the shared goal of getting the robot reach a particular user-selected goal position which is defined in an environment for which there is already a global map available and to which the robot has access. The global map is used for the localization of the robot and is the reason why we have the possibility of pre-defining goal positions in the robot's workspace. The robot is not aware of the positioning of the goals which explains the rationale for using APF as the local path planner.

Therefore, the navigation of the robot towards the goal position is a collaborative task done both by the user, who is responsible for supervising the interaction and giving linear as well angular velocity commands, and the robot being responsible for managing obstacle avoidance using its local sensory readings, while being assisted by a global predictor for the goal the user is trying to guide the robot to. The assistance system is aware of both the positioning of the goals as well as the positioning of the robot in the environment and will be described in chapter 4.

1.7 Structure of the thesis

In chapter 2, a literature review will be provided, along with the localization in chapter 3. In chapter 4 the goal assistance system will be elaborated, followed by the proximity grid and the APF in chapters 5 and 6 respectively. In chapter 7, the experiments both in the simulation and the real world along with the experimental results will be given, followed by the conclusion in the last chapter.

1.8 System Overview

In this subsection we aim to provide an overview of the system.

The robot through its laser is providing the source for the repellors and attractors, along with its pose info that will be subscribed to by the assistance system that takes as input the global goals and the user velocity commands as well. Shared navigation represents the node that based on the data coming from the repellors and attractors, compute the final APF velocity. This velocity can be combined with that coming from the user as a final control phase. It's also worth noticing that there is no direct link between the user input and the robot and also, there is no direct link between the shared navigation node and the robot either. This suggests that any velocity has to first go through the *filter* of the final control, to be properly weighted before being published to the robot.

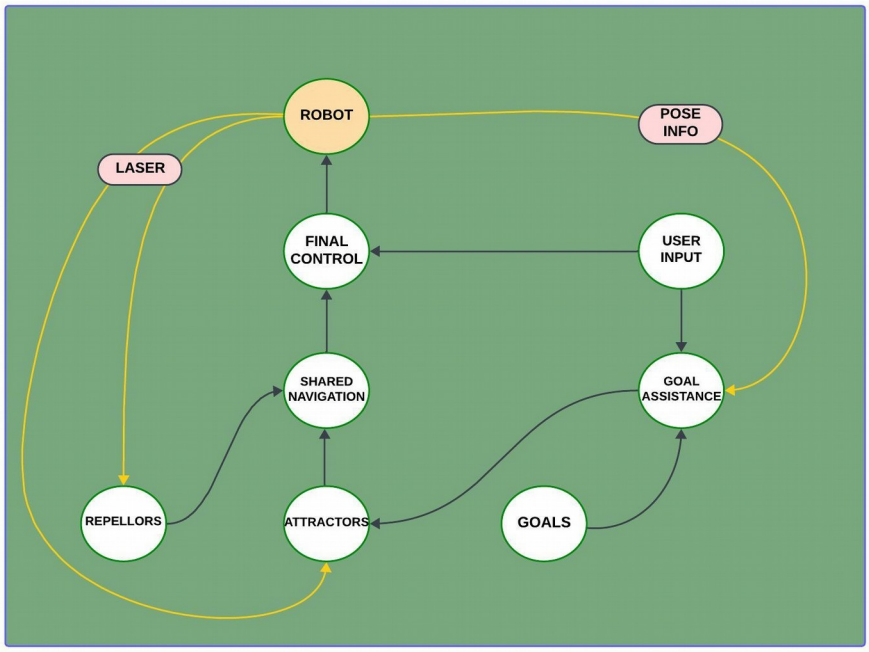

Fig. 10. The scheme of the work

2. Literature review

In this chapter, a literature review of the recent advances in shared methods and artificial potential fields will be given.
The work on this thesis started inspired from [11] in which there is a shared control teleoperation framework based on the generation of artificial potential fields around the objects. They try to tackle the problem of the APF getting stuck in local minima by introducing the concept of dynamically generated *escape points.* Escape points are cartesian points that generate pathways for the robot to follow in order for it to avoid having a collision. This is done by a procedure in which they first generate a set of convex vertices and their corresponding convex edges of the detected obstacles, followed by a sampling of the escape points by solving a constraint satisfaction problem. Moreover, it's interesting to mention that the attractive field is generated both around the target(s) and around the generated escape points.

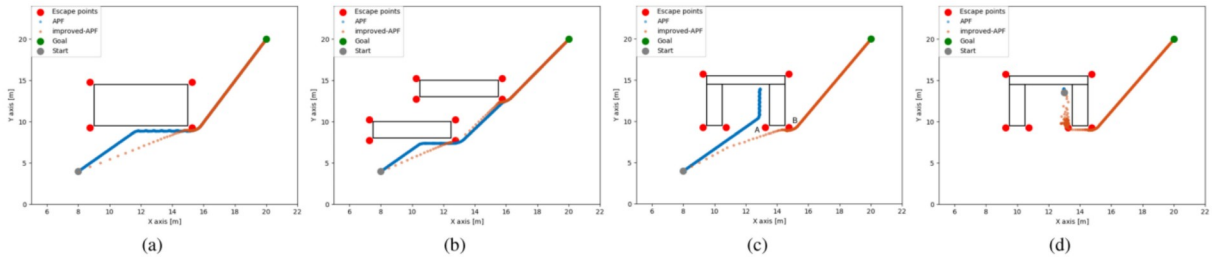

Fig. 11. The generation of escape points can prevent the robot from getting trapped inside the local minima. Moreover, if the robot starts moving from inside a local minimum, escape points will guide the robots outside while, the standard APF, without the escape points, is unable to do so [11]

In the figure above, (a) and (b) visualize a situation in which there is no local minima and the robot is successfully able to reach the target (red dots) using the APF. In fact it's also exposed that the escape points (orange dots) are able to guide the robot on a path that will make it reach the goal earlier. Instead in (c) and (d) the obstacles are forming a hollow space in between which will cause the APF to get stuck in local minima. In (c) the blue line is the path planned by the APF which gets stuck while the escape points have generated a safe path to the goal. In (d) the starting point of the robot is inside a local minimum and the escape points are still able to take the robot out of this position and guide it towards the goal.
More on shared methods, an interesting work was presented in [12] in which they introduce a framework for shared-autonomy teleoperation of telepresence robots, driven by a people-aware navigation. The telepresence robots can keep people

company and assist them, especially in the case of their work which was during the COVID-19 pandemic. These robots apart from being able to receive user input for motion and for approaching their targets who are also humans, should also respect the social manners for instance keeping an appropriate social distance from the other people in the way who are not selected as a target. The robot is basically driven by two main forces namely *protective behavior* and *goal-oriented behavior.* The former is generated in the vicinity of obstacles in the environment and also the people who are not targets. While the latter is generated around the person whom the user has selected as the target. The final motion of the robot will be influenced by both of these forces.

In their results they report that in the shared modality of teleoperation, there was a rather significant decrese in the user's workload as the number of the commands that the user would have to send was almost three times less than that in a pure teleoperation modality. However in the shared modality there is an increase in the level of reliability of the user to the system since the system would have less collisions compared to the pure teleoperation.

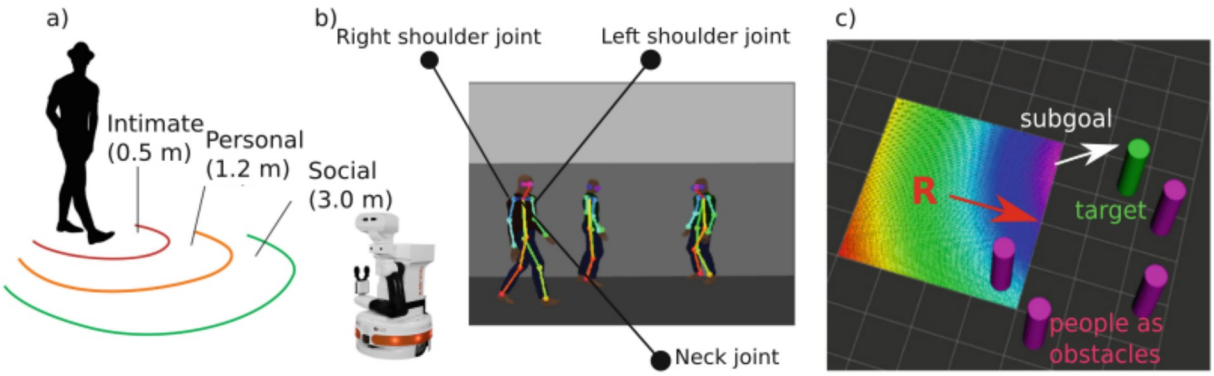

Fig. 12. (a) The robot should be aware of social manners if it's supposed to be interacting with humans socially (b) Recognition of humans (c) The robot is exposed to the fusion of attractive and repulsive forces and that's how it will be guided far from the obstacles and towards the target(s) [12]

In the figure above, in (a) it's shown that the robot should consider the boundaries when approaching people. The personal space can be divided into three zones of intimate, personal and social. In (b), there's a figure containing three detected people from which the system will estimate their diretion and position based on the estimation of their left and right shoulders and neck joints. In (c), the robot is shown being exposed to a fusion of the attractive and repulsive forces. The people who are considered as an obstacle are represented as purple pillar while the target is in green.

The subtarget in white is a temporary goal for the robot based on the fusion of the attractive and repulsive forces into a heuristic function.
Another work [5] exploring the realm of telepresence robots as well, focuses on the brain-machine interfaces (BMIs) in order to receive the user commands. They explain the shortcomings of these interfaces as they are prone to noise and error and the fact the impaired user using them, has to make a lot of mental effort in order to control the robot. Their contribution is a shared-intelligence framework in order to fully exploit the functionalities of an intelligent robot in this scenario.
The rationale for using a shared-intelligence frame instead of a common shared-control one, is to be able to implement the user's actions in a safer way. The intelligence of the robot lies in the fact the robot will decide its next motion not only based on its sensory readings from the environment and the user's input but also based on the *natural direction of robot's motion.*
As a result of testing the framework on 13 healthy people and it turned out that the robot's motion is in line with the intentions (through the BMI) of the user. It can be said that their major finding highlighting the potentials of the shared-intelligence methods, was the fact that the robot's motion through the BMI was coherent with that from the continuous teleoperation of the robot.
The human-robot interaction was further explored in [10] by focusing on a trade-off between human safety and robot efficiency. Explaining that when the robots and humans are put together in an environment to interact with each other, the safety for the humans has to be strictly prioritized and that's when the efficiency of the robot might get jeopardized.
More specific to their case, they consider a system in which there are multiple autonomous mobile robots interacting with humans and the path planning of the robots is based on the generation of potential fields. The repulsive potential field is generated uniformly around static obstacles based on their *position*, a condition that can not be of much use when it comes to an environment with dynamic obstacles. Moreover, a uniform circular repulsive field can obstruct the robot's path towards the goal which might ultimately lead to the robot not being able to reach the destination. Their contribuition is the introduction of a behavior potential for dynamic obstacles. The behavior potential is generated based on not only the position of the object but also its velocity and direction of movement. Furthermore, this potential is capable of changing its shape according to the obstacle, as opposed to the usual repulsive potential field that constantly appears in a circular shape around the obstacle and thus repelling the robot equally all around the object.

The repulsive force based on a behavior potential field is reduced as the velocity of the object is reduced. Suggesting that this type of potential field won't be generated for static objects, as for those the typical circular repulsive field will be used.
In the figure below, there are three different scenarios of repulsive potential fields. In all of them, the robot is moving from right to left on a straight line and the human from the left to right. In (a), we see the typical repulsive potential field around the moving human with its peak being located right at the center of the field; where the energy is maximized. In (b) and (c) the behavior potential fields are provided for the human in which the amount of the elongation is based on how fast the human is moving, in (b) for instance, the human is moving rather slowly as opposed to having a faster movement in (c) and thus a longer field. If the human is moving more fast, the field will be bigger and thus the robot will be affected by the repulsive forces from a longer distance and will approach the moving human more carefully and could potentially acquire some obstacle avoidance behavior based on this elongated field. While with a smaller potential field, the robot will get closer to the moving human before starting to avoid it as an obstacle avoidance measure.

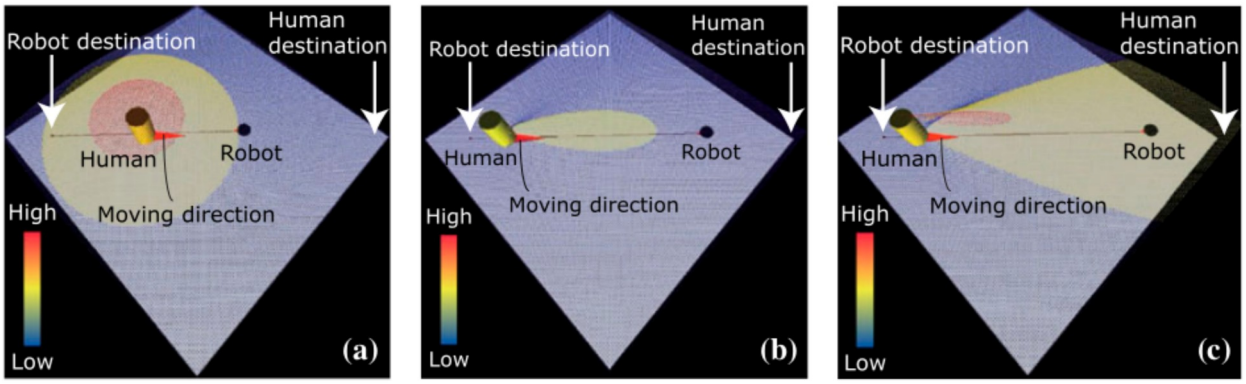

Fig. 13. (a) A typical circular repulsive field (b) behvaior potential field generated for a human with a slow pace (c) behavior potential field generated for a human with a faster pace [10]

They proceed to elaborate the problem of *global robot congestion* which stems from the fact the robots moving in an environment, populated both by humans and robots, are affected by multiple repulsive potential fields coming from different obstacles. The shape of the total repulsive field will thus have several peaks (each obstacle being located at the peak, having the highest amount of repulsion). This will cause the robots, moving on the gradient direcions from these peaks resulting in an uncoordinated motion and finally the congestion of robots from a global point of view.

In order to tackle this problem, they provide a second potential field namely the congestion potential field. The congestion potential field contains a single peak and is based on the position of the robots with respect to the origin of the system. The goal is for each robot to be able to know the density information of a position, a factor that will influence the path planning of the robot along with the previously mentioned repulsive potential fields.
The figure below depicts a situation in which the global robot congestion occurs due to the existance of several peaks, each for either of the repulsive fields. This could cause the robots in adjacent peaks, be affected by different repulsive fields in the opposite directions. As a consequence, the robots will collide at the trough of the combination of fields.

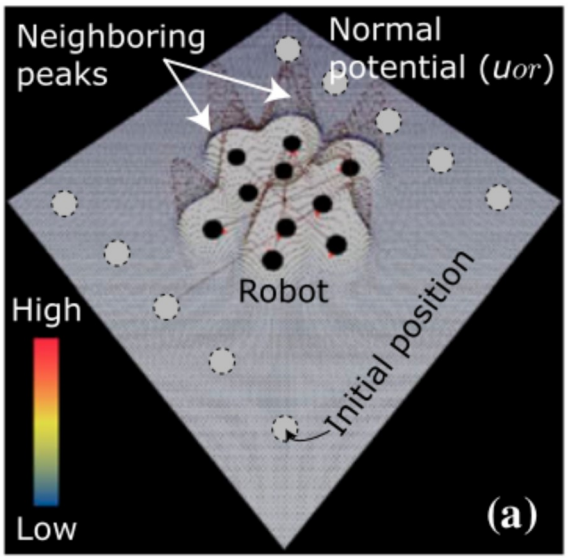

Fig. 14. Adjacent repulsive fields [10]

The path planning can be approach from a fuzzy point of view [9]. Given the usual case of a mobile robot navigation task, the robot has to navigate in an unknown environment based on its sensory readings and act accordingly in real time. All of these make APF a proper online path planning technique since an online algorithm does not require any information about the environment in advance and also, APF is a low cost and easy technique.
As we mentioned previously, APF is prone to the local minima problem. The algorithm might lead the robot towards a situation from which the robot can not escape from or in the case of the robot starting the motion from inside of a local minimum, APF can not find a path for the robot to get out of the situation.
Due to the inherent nature of the case of an autonomous mobile robot navigation which deals with a vast amount of imprecise and uncertain data perceived from the

environment, the fuzzy approaches in this field can be considered as close-to-perfect solutions.

The problem of the local minima is tackled through a fuzzy logic modification of the approach introduced in [13]. The algorithm is executed whenever the robot is trapped in a local minimum. The distances l_1 and l_2 between the robot and the two extreme points of the obstacle are calculated. The introduce a *virtual plane* which connects the two exterme points and creates a plane. The angle between l_1 and l_2 is computed and later the robot will be attracted to a *virtual new objective* defined along the angle bisector of the aforementioned angle. The robot, attracted to this target, will pull itself out of the local minimum. Moreover, the robot will not be attracted to fall into the local minimum again since the virtual plane will now be generating a repulsive force. The setup of the algorithm is illustrated in the figure below:

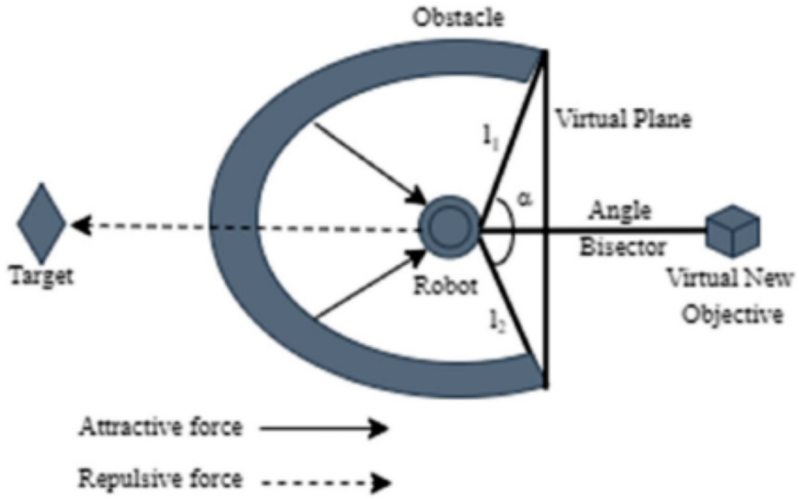

Fig. 15. The robot will be attracted towards a newly defined virtual target while it's being repelled by the exterme points on the obstacle as a way of reassuring that it won't fall into the local minimum again [9]

The fuzzy based algorithm makes use of a fuzzy based control which takes as input several parameters such as the velocity error between the current velocity of the robot and the computed velocity based on APF, plus the error angle between the actual heading orientation of the robot and the one computed based on the net APF force. The output of the control is the angular velocity of the robot.

Ultimately, the overal setup of the system includes the path planning based on APF which is based on the generation of attractive and repulsive forces and the execution of the fuzzy block of the algorithm in case of the robot getting trapped in local minima. In the figure below, the path planned by the APF in the simulation is depicted as blue:

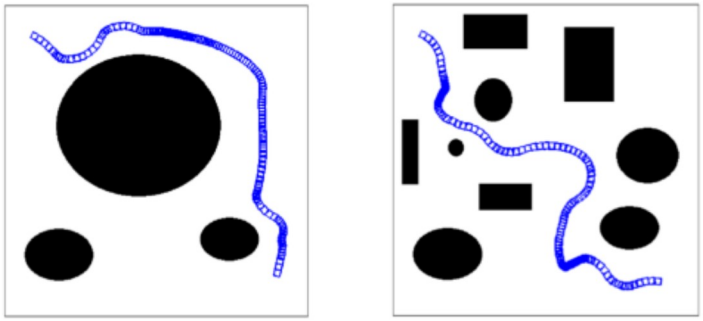

Fig. 16. Path generated by APF [9]

In another figure, the path planning based on the fuzzy approach is shown:

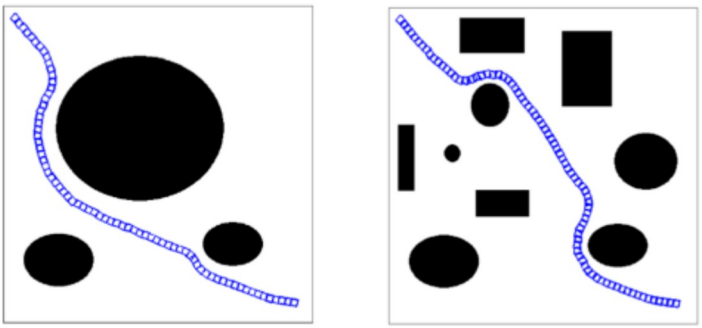

Fig. 17. Path generated by the fuzzy block [9]

As mentioned in the previous chapter, heuristic methods for path planning are usually more flexible when it comes to uncertainties and are less computationally expensive compared to the classical methods. Although in the recent years, the research and reports on heuristic methods have been mushrooming in the literature, classical methods, such as the utilized APF in this very thesis, are still in common use.
One of these heuristic methods are the artificial neural networks (ANNs). In the case of a mobile robot navigation, the inputs will be the sensory readings of the robot and the outputs will be the steering direction or the velocities that must be sent to the wheels. ANNs in this scenario can be used in both environment from which a previous knowledge is available and also when such global information is not present.

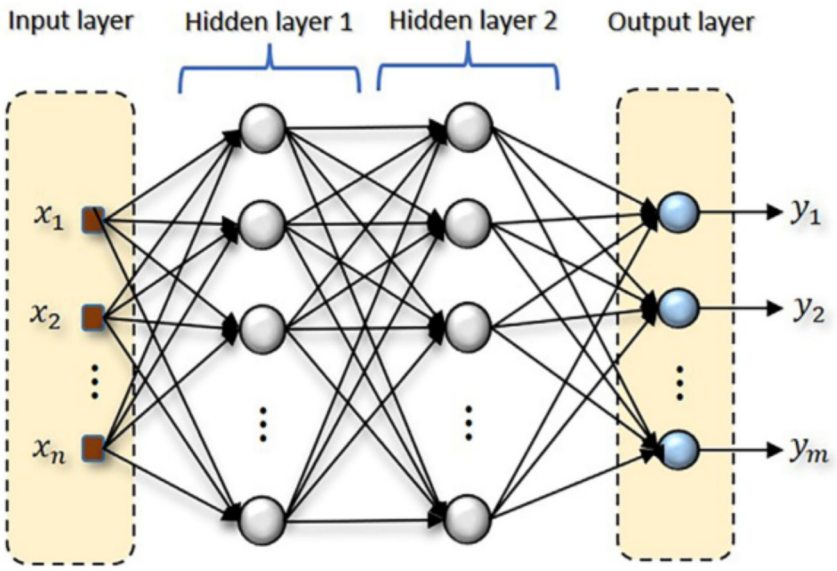

Fig. 18. Typical ANN with two hidden layers [6]

3. Localization and mapping

One of the crucial constituent technologies for autonomous mobile robots is SLAM (simultaneous localization and mapping), suppoting environmental recognition and path planning.
Solving SLAM requires solving two subproblems of localization and mapping. In localization the aim is for the robot to estimate its own pose in the map and in mapping the objective is for the robot to create a map of the sensed environment. This is challenging since either of the two, depends on the other. In order words, in order to solve mapping, we need information from localization and vice versa.
G-mapping is one of the famous SLAM methods for creating occupancy grids of the environment to represent its map. Occupancy grid is a grid-like structure in which every cell stores a probabilistic estimation of whether that cell in the environment is occupied or not.
Furthermore, self-localization is an important feature which mobile robots should be equipped with in order to fulfill tasks such as navigation. Localization is either global or local. In local localization, the initial position of the robot is known and the robot uses its sensory readings to track and update its own position. Common methods are odometry and inertial navigation methods.
In global methods instead, it is assumed that the initial position of the robot is not known and there should be some external information directly provided by the robot's sensors to estimate its pose.
The state of the art method in localization is the Adaptive Monte Carlo Localization which is a probabilistic algorithm [14]. It works by estimating a normal probability distribution representing the robot's position at every time instance as a set of hypotheses. It introduces the concept of a *particle* which is basically a tuple in the format <x, w, h> in which x is a hypothesis, w the dedicated weight to that hypothesis and h being the indicator of the likelihood of each particle with regards to the last perception of the robot.
The AMCL algorithm has three phases in which the particle tuples are updated. In other words, in a sequential manner the position of the particles and the weights of the particles are updated. In the end, based on the weights of the hypotheses, the hypotheses having a low weight are removed and replaced with new hypotheses that are created near to the ones with high weights. The algorithm adapts the number of particles to the uncertainty of the set of hypotheses. That is how concentrated or dispersed the particles are. When the uncertainty is low and the particles are more

concentrated, the number of particles is reduced and instead when the uncertainty is high and the particles are more dispersed, the number is increased.

The key advantage of AMCL is probably its adaptability to environmental changes. As the robot navigates through the environment, the particles are adjusted to reflect the changes in sensor readings and robot motion. This adaptability gaurantees that the robot's estimate of its pose remains accurate even in the presence of uncertainty. In the figure below we see a scenario in which a bunch of tiny red arrows, which happen to be the particles of AMCL, are in a rather dense which basically indicates a more certain estimation of the pose:

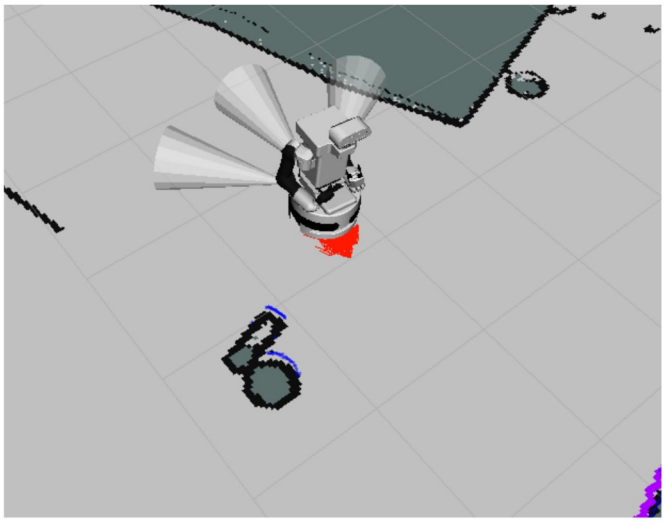

Fig. 19. Dense particles

And instead in another figure below, there is a less dense area containing the particles and thus there is a higher uncertainity regarding the pose:

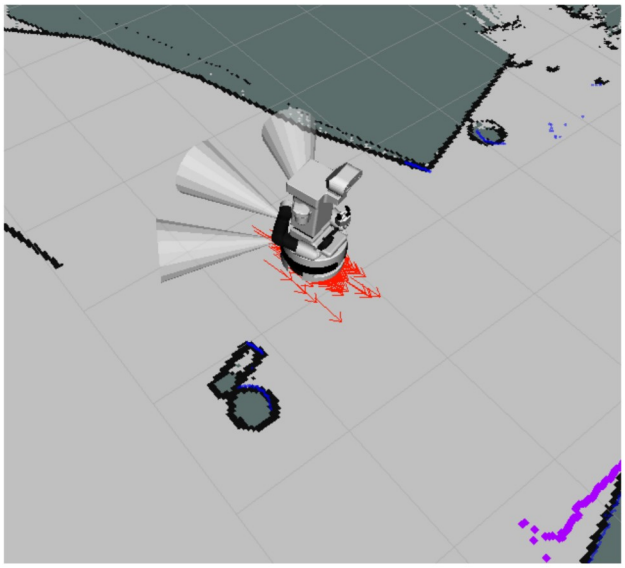

Fig. 20. Rather widespread particles

It’s common for the localization to get *messy* right at the start of the motion of the robot. As the robot keeps progressing, the particles follow up and the estimation gets more accurate.
The red arrows are constantly pointing towards the direction the user is guiding the robot to. For instance, in the figure above the direction of the motion is towards the right side of the figure.
In the figure below there is a good case in which the mapping is done successfully. The black thick lines represent the boundary of the map and the blue lines are the laserscan of the robot being stopped at the encounter of an obstacle. A good case of mapping is the one in which there is an acceptable matching between the sensory readings and the known map of the environment.

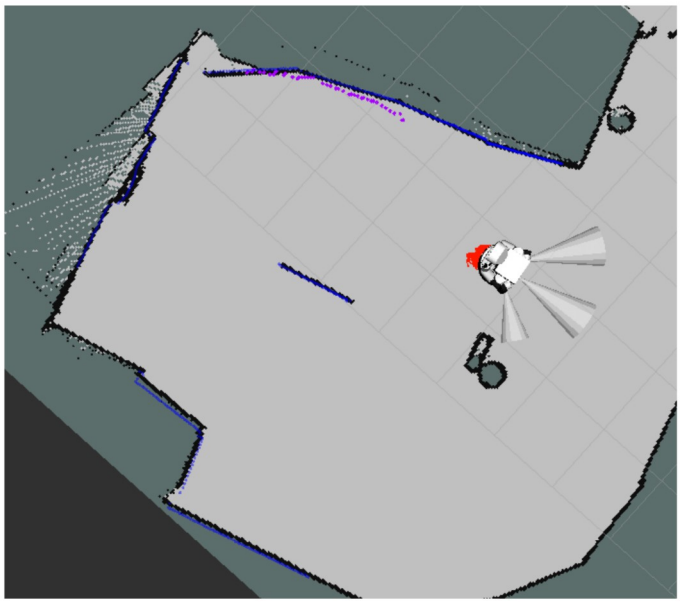

Fig. 21. Acceptable localization

While below is an example of the mapping not being done successfully as there is an apparent mismatch between the sensory readings and the map:

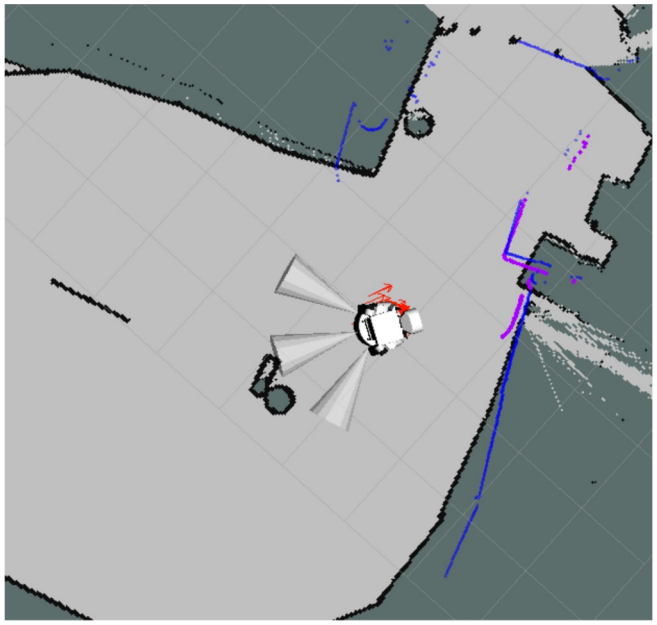

Fig. 22. Fast linear motion of the robot might sometimes make the robot disoriented with respect to the known map

It was often observed that in the case of an unsuccessful mapping, sending a sequence of angular velocity commands would mitigate the problem:

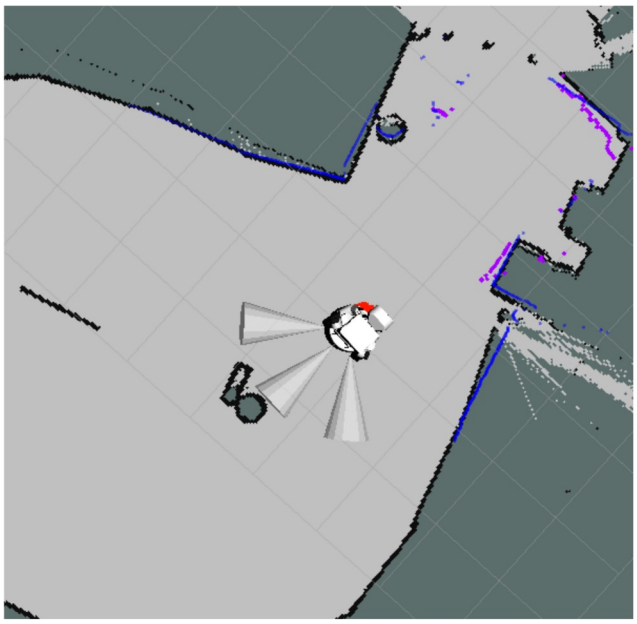

Fig. 23. Problem of figure 22 gets resolved by a sequence of angular velocity commands

Moreover, it can be see that whenever there is a poor mapping, the localization also ends up not being good enough as the area containing the particles becomes less

dense. This explains the dependency of mapping and localization on one another that was previously mentioned.

4. Goal assistance

The goal assistance system used in this work is the one introduced in [11]. Having a known number of goals (3 in our case) and the assumption that throughout the interaction the user will be guiding the robot towards only one goal while being aware that the robot is handling the obstacle avoidance matter, it's possible to predict the probability of either of the goals based on the series of user inputs. In the current version of the assistance system, only a non-zero linear velocity command coming from the user will invoke an update of the probabilities and thus angular velocity commands are ignored in this scenario.

A Markov decision process (MDP) is used for the modeling of the user's behavior. The MDP is constructed by states, each representing a robot position p in its workspace. It has been defined a goal state p_g , a transition function $T(p'/p,u)$ with u being the user's action, policy $\pi(u/p,p_g)$ and a cost function $C_g^u(p,u)$ being defined as:

$$C_g^u(\boldsymbol{p},u)=\begin{cases}\alpha, & d>\delta\\ \alpha\frac{d}{\delta}, & d\leq\delta\end{cases}$$

Eq. 7.

with δ being a threshold used for determining whether the robot has become too close to a goal position and d being the distance between the *next* robot's position and p_g . By having the optimal user's policy plus a sequence of user's inputs and robot's states $\xi^{(0\to t)}$ from the beginining of the interaction until the current time, it's feasible to compute the probability that the user is trying to get the robot to reach the specific goal position p_g when p_g belongs to a set of known goal positions as below:

$$P(p_g)=p(p_g/\xi^{(0\to t)})=(p(\xi^{(0\to t)}/p_g)p(p_g))/(\Sigma_{p'_g}p\,\xi^{(0\to t)}p(p'_g))$$

Eq. 8.

The assistance is made up of two service servers, one for the initial configurations such as initializing the parameters that will be received and sent back as part of the communication between the robot and other server which will be receiving a request containing the received user's velocity command and the current pose (position and orientation) of the robot. In the response service file, there will be provided two important informtion; one being the updated probability distribution and the other

being the id of the goal having the highest probability (since the assistance system already knows the goals and their dedicated identifier number).
The probability distribution will be stored in the form of a vector from which the probability of either of the goals can be accessed and later used for computing the *center of mass* of all the goals. This will be discussed further in chapter 5.
It's worth mentioning the challenge we faced in tuning the parameters of the predictor such as the δ since the position of the center of mass should not change more slowly than the robot changing position. In other words, the computed center of mass is further from the target than the robot is from the target, there is basially no use in using the assistance. If δ is chosen as too small, the probability of the desired goal won't increase fast enough because the robot has to get super close to the target in order to receive the assistance he/she needs. We remind that these probabilities will later be used as weights of each goals to compute a center of mass of the goals, and if the probabilities are not well-adjusted with the behavior of the robot, the assistance won't be much of a help. As a rule of thumb, δ should be 2 or 3 times bigger than the width of the robot (given that the robot has a circular base).
On the other hand, if the δ is chosen so big, the assistance would be so sensitive to the behavior of the user. Any single command can potentially make a dramatic change in the probabilities and this was strongly discouraged since neither the simulated world on Gazebo and nor the real world were huge spaces.
The same care had to be given to α . It deals with how much each user action has to be penalized based on distance. Having a larger alpha, could potentially increase the sensitivity of the assistance in the sense that each action towards one specific goal could cause the probability of that goal increase in a more satisfactory way.

5. Proximity grid

In this thesis the concept of a proximity grid [15] is used as a tool to measure the vicinity or proximity of the robot to its surroundings. The proximity grid can be thought of as a virtual grid made up of several sectors, each sector holds a range of angles. The grid is constantly surrounding the robot and moves with it.
Each sector of the grid has a unique corresponding ID which is used as an identifier for that specific sector. Moreover, there is also a designated value field for every sector which holds the euclidean distance from the robot to the first object that blocks the laserscan ray, emitted from the base of the robot.

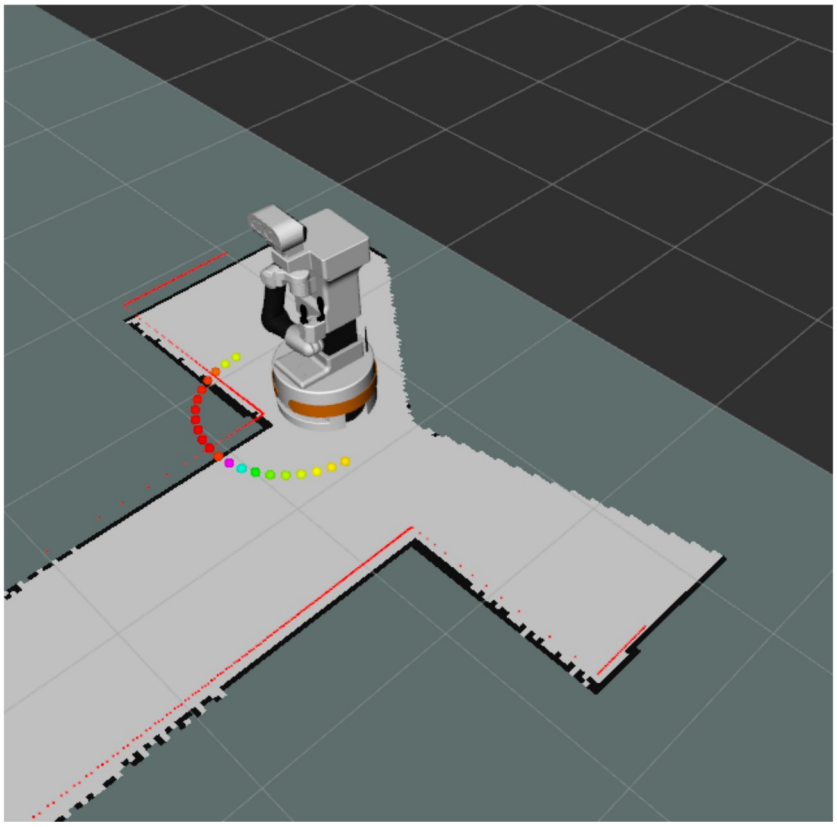

Fig. 24. Proximity grid around TIAGo

In the figure above, there is the visualization of TIAGo on RViz [16]. The black bold lines framing the white region in which the robot is located, are defining the boundary of the known map and therefore indicate the existence of a wall. The red thin lines in parallel to the thick black boundaries, make up the sequence of points on which the laserscan rays of the robot is being blocked by the walls.
The proximity grid is made up of the tiny colorful spheres mainly at the front of the robot. Each sphere represents a single sector in the proximity grid. The difference in

color actually indicates the difference in the measured distance between the robot to the closest object. The red sectors are faced towards a region in which the distances are relatively small and thus the probabiltiy of a collision is higher in comparison to the other sectors. Moving in a gradient of colors from red to green, blue and purple we encounter regions which suggest a progressively *safer* space for the robot to move towards.

The proximity grid is configured to be spanned in an angle range of (-90, 81.42) degrees and having an angle difference of 9.0 degrees between any two sequential sectors which will make up 21 sectors. Moreover, there are min and max ranges defined as well in meters, with the min range being 0.0 and the max being 6.0 meters. The ID of the sectors start from the leftmost sector and keeps increasing one by one. For instance the sector with ID of 0, is situated between an angle range of (-90, -81) degrees.

In this work, two separate instances proximity grids are utilized to assist the navigation of the robot. The information provided by these two grids are published on two distinct topics in the ROS network, namely `/repellors` and `/attractors` containing the data regarding the repulsion from obstacles and the attraction towards the goal respectively.

The way the two proximity grids are filled with data is essentially different. In the case of repellors, every single sector of the grid is filled with the shortest distance information but concerning the attractors, only a single sector in the grid gets filled with data and the rest are *inf* values; suggesting a sector with no significant information. This single sector is supposed to be pointing in the direction towards which the robot should be attracted.

Computing the shortest distance values from the robot to the obstacles in the case of filling the repellors grid is a trivial task since these distance values are already provided in the `/scan_raw` topic, published as soon as the simlutation of the robot in the world starts.

However, the case of the attractors is distinct since the only non-inf sector is filled with the shortest distance value between the robot and the *center of mass* of the goals. The predicted probability for either of the three goals are being published on the `/final_goal` topic and are later used as weight parameters in computing the center of mass using the below formula:

$$center\ of\ mass = ((goal_1 x\ weight_1) + (goal_2 * weight_2) + (goal_3 * weight_3)) / (weight_1 + weight_2 + weight_3)$$

Eq. 9.

and since the weights add up to 1 the formula is simplified to:

$$center\ of\ mass = (goal_1\ x\ weight_1) + (goal_2 * weight_2) + (goal_3 * weight_3)$$

Eq. 10.

Using the center of mass as the *subgoal* of the navigation is a way of having the robot being attracted to all the goals at the same time especially in the beginning when the uncertainity in predicting the actual goal is higher.
Moreover, the center of mass is constantly getting updated as the robot *rotate* towards it and the probability of the desired goal keeps increasing. Therefore, as the shared control navigation progresses, the center of mass keeps getting updated to a more precise approximation of the actual user's goal.
The next step is to find the correct sector in the proximity grid of the attractors which is *sufficiently* headed towards the computed center of mass. The information we will be using is the position of the center of mass/target:

$$angle = \arctan^2(target_y, target_x)$$

Eq. 11.

The angle computed above is the one between the x-axis and the euclidean distance (shortest distance) between the robot and the center of mass. The x-axis in question is the one in the heading direction of the robot, pointing outwards at front. By finding the aforementioned angle, we are realizing the amount of rotation the robot should do in order to be properly face the goal. The scenario we deal with is similar to the figure provided below, where G denotes the current subgoal, R indicating the robot and theta being the angle we seek to calculate:

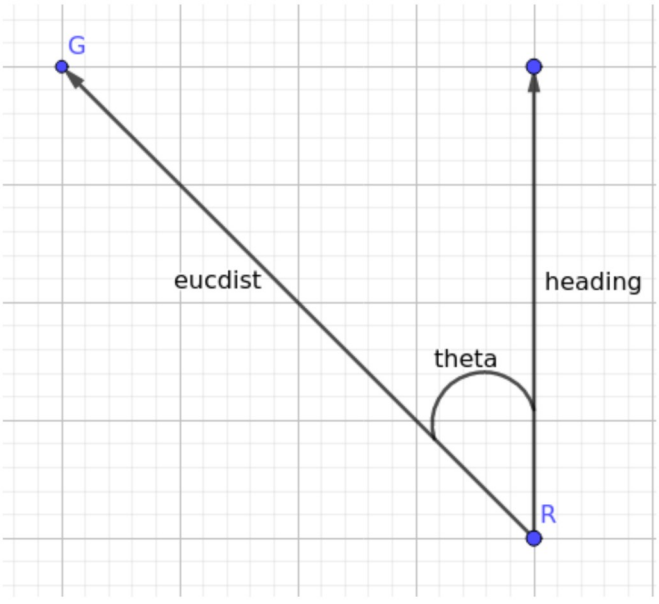

Fig. 25. Theta is the angle between the heading direction of the robot and the vector representing the euclidean distance between the target and the goal

After having computed the angle between the heading direction of the robot and the eculidean distance between the robot and the target, it's time to fill the one sector in the attractors proximity grid which points to the target. This is done by finding the id of the grid in which our angle lies in using the method `GetSectorId(angle)` which is one of the functions implemented in the dedicated class of the proximity grid. Having the id, we are able to fill the corresponding sector of the grid using another class methods of the proximity grid, namely `SetSectorByPolar (angle, distance)`, which having the angle and the euclidean distance from the goal, fills the sector with the provided euclidean distance.

There will be an attractive force towards the goal, generated by the last non-empty and non-infinite sector of the grid, which since there will be only one sector in which our computed angle lies in, there will be only one non-empty and non-infinite sector available. The target will be the angle corresponding to the sector which points to the goal and will thus guide the robot towards the goal.

Lastly, it might happen that the computed angle between the heading direction of the robot and the vector representing the euclidean distance from the goal falls outside of the grid. In other words, the angle is either smaller than the minimum angle or is bigger than the maximum angle in the grid. In the first case, the angle is smaller than -90.0 degrees and in order to have the goal fall in the angle range of the grid, the robot starts rotating anticlockwise and for the second case, the angle is bigger than 81.42 degrees and the robot starts moving clockwise to have the goal fall in the grid.

In the link below, there is a video of the goal being outside of the grid and the robot rotating in order to make the goal fall in the grid:

6. Fusion of the potential field and user commands

There should be a mechanism (or a node in the ROS network) fusing the effect of the user command and the artificial potential fields and apply it on the robot. This step is necessary since the navigation is done in a shared manner and both the robot and the user are having the same goal.
It's important how this fusion is done both for safety reasons and for the sake of giving the user a sense of *being in control*. As mentioned in chapter 4, the user sends the velocity commands knowing that the robot is handling the obstacle avoidance. This indicates that the user is fully focusing on leading the robot towards the desired goal without having to worry about the collisions he/she could cause.
First of all, we will be reviewing the formulas dedicated to the velocity computation.
As discussed in the introduction of this thesis, potential fields are defined as potential functions. Choosing the right function as the potential is not a trivial task and has to be done with regards to the application it's being used for. For the case of our work, the following function is used for the generation of a repulsive field around the obstacles (also called the FIRAS function):

$$U_r(\boldsymbol{p}) = \begin{cases} \frac{\eta}{2}\left(\frac{1}{\rho(\boldsymbol{p})} - \frac{1}{\rho_0}\right)^2, & \text{if} \rho(\boldsymbol{p}) \leq \rho_0 \\ 0, & \text{if} \rho(\boldsymbol{p}) > \rho_0 \end{cases}$$

Eq. 12.

And the repulsive force:

$$\boldsymbol{f}_r(\boldsymbol{p}) = -\nabla U_r(\boldsymbol{p}) = \frac{\eta}{\rho^2(\boldsymbol{p})}\left(\frac{1}{\rho(\boldsymbol{p})} - \frac{1}{\rho_0}\right)\nabla\rho(\boldsymbol{p}).$$

Eq. 13.

ρ represents the distance between the robot (current robot state of p) the closest point on the obstacle. While ρ_0 is the limit distance of the potential field influence (also called the influence ray of the obstacle or the largest impact distance of the obstacle) indicating the furthest distance on which the repulsive field of an object will stay effective. [9]
In case of the attractors, the potential function generating an attractive field around the goal, is defined as:

$$U_a^g(\boldsymbol{p}) = k_a \|\boldsymbol{p}_g - \boldsymbol{p}\|$$

Eq. 14.

And the attractive force:

$$\boldsymbol{f}_a(\boldsymbol{p}) = -\nabla U_a^g(\boldsymbol{p}) = k_a \frac{\boldsymbol{p}_g - \boldsymbol{p}}{\|\boldsymbol{p}_g - \boldsymbol{p}\|}.$$

Eq. 15.

Having the gradient of the two fields, we have the velocity of the robot affected by a combination of the attractive and repulsive forces. Since our case is a shared control scenario, we should also take into account the user velocity commands and thus we have as the final velocity sent to the controllers of robot:

$$\boldsymbol{v}_r = \boldsymbol{v}_{\text{in}} - (\nabla U_a^g(\boldsymbol{p}) + \nabla U_r(\boldsymbol{p})).$$

Eq. 16.

The formula provided above is based on the assumption of having only one goal that could be tracked by the user. In this thesis work, we have defined 3 goals and towards either of them the user might be leading the robot. Thus, the predicted probability for all the goals must be taken into account when computing the final velocity of the robot:

$$\boldsymbol{v}_r = \boldsymbol{v}_{in} - \left(\sum_{\boldsymbol{p}_g \in \mathcal{W}} P(\boldsymbol{p}_g) \nabla U_a^g(\boldsymbol{p}) + \nabla U_r(\boldsymbol{p}) \right)$$

Eq. 17.

The velocity imposed by the APF is made up of a linear and an angular part. The effect of the APF will be mainly *observed* through the angular motion of the robot. This is due to the fact that the current mechanism for fusing the user command and the APF, has a gate-like behavior with the user input. The robot will only start moving forward if there is a non-zero linear velocity command received from the user. In this scenario, the final linear velocity of the robot will be a weighted average of the two linear velocities. The linear velocity of the APF is almost always non-zero except for a situation in which the robot is too close to an object. Often times, a higher weight is given to the APF due to safety reasons.

The angular motion due to the APF, is in a way that the robot should be put away from the obstacles and also being roughly headed towards the center of mass of the goals. The first behavior is achievable for the case of dyanmic obstacles as well. If an object that has not been in the map is suddently put at the front of the robot, the repulsive field generated around the new object will successfully cause the robot to steer away from the obstacle.
There have been defined the strength value parameters for either of the two forces, indicating the strength they will have in creating repulsion and attraction. Finding a good balance between the two forces is rather challenging but it must be considered. If a high strength is given to the attractors, the robot might end up forcing its way to pass through the walls in order to reach the goal. Instead, if a high strength is given to the repellors, the robot might start having troubles in navigation since everything in the environment no matter how far they are, will be dramatically repelling the robot. The peak of such behavior can be seen in narrow spaces where the robot will be heavily oscillating itself through the free space.

7. Experiments and results

7.1. Robotic platform

The robotic platform used for this thesis was the TIAGo++ from PAL Robotics [1]. The robot has a differential drive base with two wheels that can move either forward or backward with the both wheels having the same velocity and in order for the robot to perform an angular motion, the robot has to rotate in place which is feasible by having the both wheels turning, given that one wheel is turning faster than the other. Although these low-level details are for sure not managed by the user.

Moreover, the robot is equipped with a laser emitted from its base which was used for navigation and obstacle avoidance. There's also an RGB-D camera on the head of the robot which is often used in manipulation tasks and also sometimes for obstacle avoidance when the obstacles can not be perceived by the laser. However, in this work the last feature was not explored.

The robot was often controlled by either a wireless Logitech joystick controller which was able to control the head, torso, end effectors and the mobile base or simply the keyboard of a PC connected to the robot. The latter option was utilized in this project to teleoperate the robot.

Fig. 26. TIAGo [1]

7.2 Deployment of the code

In order for us to be able to execute the code on the robot, the ROS workspace which was used for developing the project, had to be deployed on TIAGo. There were

already several deployed packages so there was no need to fully deploy every single package in the workspace. In this work, 6 packages were deployed on the robot:

- *Goals*; dealing with the definition of reachable target positions with respect to the map of the environment and broadcasting them, plus managing the communication with the assistance for receiving the predicted probabilities and publishing the center of mass of the goals which will later be used for the generation of the attractive field.
- *Key_input*; managing the pure teleoperation of the robot and publishing the velocity values to be received by several others nodes in the network
- *Localization*; managing the localization of the robot within the world and also with respect to its own reference frame using SLAM and publishing the pose information
- *Proximity_grid*; creating the two proximity grids around the robot based on the information received from the laserscan plus the visualization tools for the grids on RViz. The package contains modifications on the original implementation provided in [18]
- *Shared_navigation*; the main package responsible for computing the velocities of the APF based on the information provided from the *proximity_grid*, fusing them with the user input and publishing them to be received by the controllers of the robot, again the package contains modifications such as the addition of new nodes while the logic behind the computation of the APF-based velocities is left intact. Original implementation provided in [18]
- *Predictor_assistance*; the package contains both a navigation assistance which was not utilized and a goal assistance which was explored. The pacakge contains a few modifications on some of the parameters for tuning the performance of the predictor. Orignial implementation in [19]

For deploying the code, there are two options. Either to deploy only the package we have recently modified using the two commands below [17]:

```
alis deploy="rosrun pal_deploy deploy.py"
deploy -p <package_name> tiago-87c
```

Or to deploy the entire workspace at once:

```
rosrun pal_deploy deploy.py --user pal tiago-87c
```

Next would be to connect to the robot first by connecting to the Wi-Fi network of the robot *tiago-87c* and then setting the robot as the ROS_MASTER:

```
export ROS_MASTER_URI=http://tiago-87c::11311
```

and later making an SSH connection to the robot:

```
ssh pal@tiago-87c
```

Once we're inside the robot through the SSH, the ROS_MASTER will always be running and we only need to start our ros nodes.

One note regarding the *pal* in the commands: when accessing the robot there are two possible users to log on to; pal and root. For this thesis, only the pal user was used.

7.3 Experiments in the simulation

The simulator used in this work is Gazebo which is a powerful physics-based simulator. It allows a 3D representation of the robot along with the possiblity of importing desired worlds having static or dynamic obstacles. Working with Gazebo was an essential part of this thesis since it allows the user to test their code and observe the behavior of the robot in a controlled environment, prior to moving to the real world.

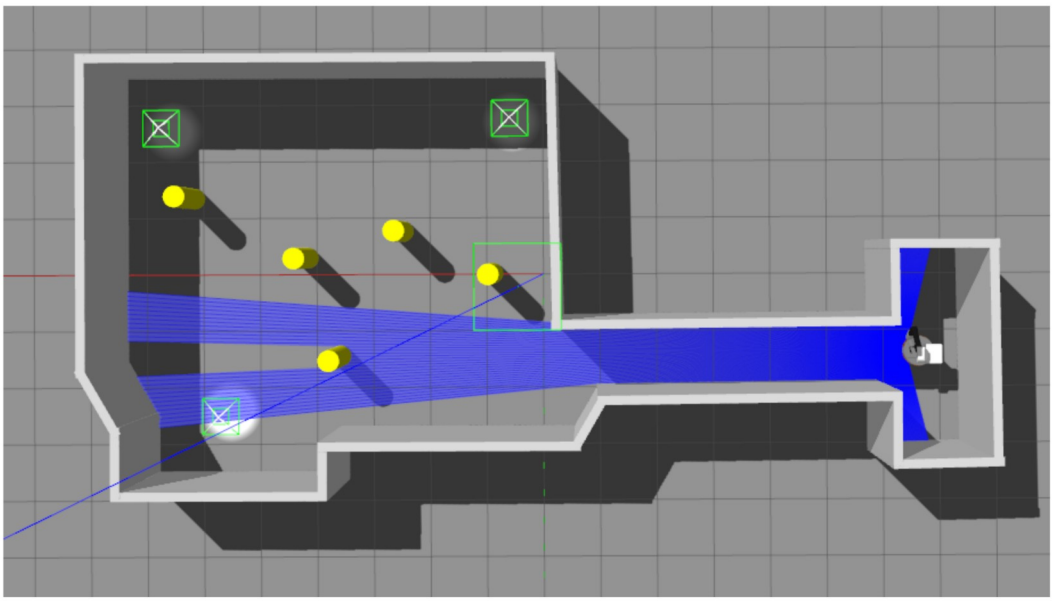

Fig. 27. The simulated world on Gazebo. The yellow cylindars are the obstacles and the targets are highlighted using green squares

The figure above depicts the simulated world that was used on Gazebo for this thesis, with the initial position of the robot being on the right of the map and three chosen goals being represented using three square lights in the corners of the large room. The obstacles are the yellow cylinders and the gray walls which are the boundaries of the room. The underlying grid of the simulated world helps us in estimating the position of the robot. The grid from which the robot starts from corresponds to the position (0, 0) with respect to the fixed/world reference frame of /map. As the robot moves forward (towards the west of the map above), it would be moving along the x-axis of the world (positive x) and if it moves to its left (south of the map above), it would be moving along the y-axis of the world (positive y).

A useful visualization tool that was also used a lot during this work is RViz which is basically an essential component of the ROS ecosystem. It allows the visualization and analysis of many features related to the mobile robot system in real-time. It makes it possible to view the data being published on specific ROS topics such as the laserscan of the robot, odometry information and transformed coordinate frames. This allows us to gain precious insight into how the robot is observing the environment along with tracking the localization and mapping of the robot in a known map. [20]

Prior to testing the real robot, experiments were done on the simulated robot in Gazebo. Experiments started with tasks as simple as teleoperating the robot using the keyboard which is quite similar to the famous turtlesim on ROS in which the user is able to move a graphical object (a turtle) on the screen. [21]

As mentioned earlier, TIAGo has a differential-drive base and is thus able to go forward and backward (linear velocity) and for rotating the robot should rotate in place about its own z axis (angular velocity). It’s also possible to combine the two velocities. The same condition applies to the simulation.

The experiments in the simulation moved further by introducing the attractive and repulsive forces to the world. Having only the repulsion from the surrounding obstacles, the robot is able to move forward without colliding with anything. In this case, the robot is basically wandering purposelessly in the world since there is no goal to generate an attraction. A recorded video of the motion of the robot purely based on the repellors is available at:

drive.google.com/drive/folders/1i_4KDME2yapR6Iqe01yC6u8LitV8jq1l

It may not be the best choice of a shared framework to not make the user feel like the robot’s behavior is not in line with what the user intends to do. But then again, when it comes to emergency situations such as that of a collision, the robot must be able to either not fall into one in the first place or to be able to *survive* the emergency

situation on its own. There is a video below in which the user tries to steer the robot towards the wall and force it to have a collision. We see how the robot is showing some resistance but not too much in order to not make the shared task too tiring or too challenging for the user, but it's also constantly trying to *escape* from the dangerous situation that the user is forcing upon it. The user in the end manages to make the collision happen but then the robot is able to survive it by pulling itself back from the obstacle:
drive.google.com/file/d/157DOAUR04XgyiOLjXOd3b1tgjFCCDi4K/view?usp=drive_link

In another attempt, a higher weight was given to the APF which results in the robot showing more resistance to the user's intention in trying to make a collision happen. In the video shared below, the robot is first faced towards the opposite side of the wall when the user tries to steer it towards the wall instead. The robot progressively shows a higher resistance until it is finally crashed into the wall by the user, followed by the robot pulling itself back from the wall and starting to steer away from it again:
drive.google.com/file/d/1LtTyhpTbVseLTQIDxtOvnyCVZKQD4ouy/view?usp=drive_link

Although it must be mentioned that (also as mentioned earlier in chapter 4) the linear velocity computed by the APF will only be applied to the robot when it is also accompanied by a non-zero linear velocity sent from the user. The videos above are created in order to simply verify the correctness of the repulsive forces while in reality and also further in the real-world experiments, the linear velocity of the APF will never be moving the robot forward on its own.
Bringing the attractors into play, generated for only one goal initially, the robot will be attracted to the goal and move towards it while avoiding the collision with the obstacles. When the robot reaches the goal position it shows an interesting pattern of motion which consists of the robot rotating around the same position which happens to be the goal. This happens due to the nature of the proximity grid of the attractors. The robot keeps rotating until the target position falls in the angular range of the proximity grid. The video is available at:
drive.google.com/drive/folders/1rQ7nSM_ir1FhfEVdQfWhzZ08VSIcsxJJ

In the link above, there are also other videos of the effect of the strength of the attractors on the behavior of the robot. If the strength is so large in value, the robot might get stuck since the attractive force is so strong that the robot wants to pass

through walls in order to reach it. In the provided videos it’s shown how the robot can get stuck at the end of the narrow pathway, on the verge of entering a bigger room while modifying the strength to some smaller value, this problem is mitigated.

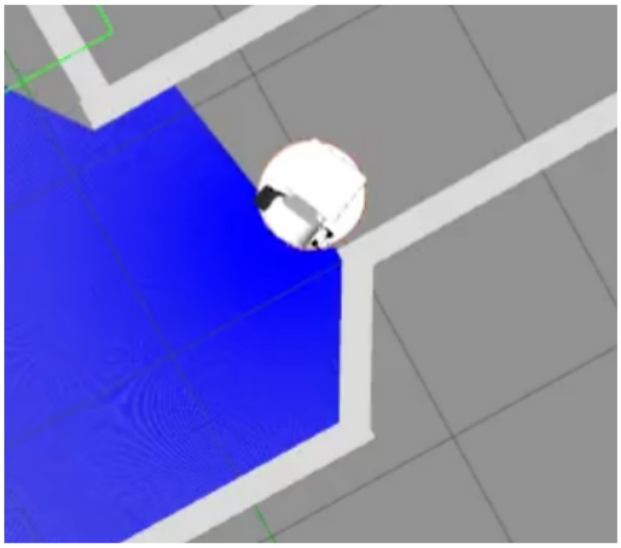

Fig. 28. TIAGo not being able to exit from the narrow hallway due to a high strength given to the attractors

In the figure above, TIAGo is stuck and is thus unable to proceed the motion. This occurs due to a large value of 0.3 for the strength of the attractors. The goal is defined on the left side of the robot as it can be seen the robot is tilted towards its left. In the figure given below instead, the strength of the attractors was reduced to 0.1 and the robot is able to exit the same narrow pathway successfully. In both of these cases, the strengh of the repellors is left intact. Moreover, in the original implementation of the artificial potential fields (from which this thesis work is inspired), the attractors have a strength as high as 1.0 while the repellors have a lower strength of 0.5. In the case of this thesis instead, the repellors always have a higher strength than the attractors, both in the simulation and on the real robot.

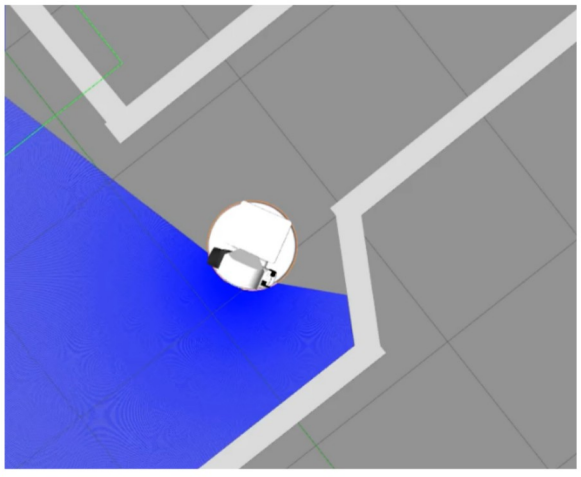

Fig. 29. TIAGo successfully exiting the narrow hallway after a reduction in the strength of the attractors

Moving forward with the experiments on attractors, the next item to test was the overal behavior of the robot, being influenced by both the repellors and attractors, when having the final target positions. The setup of this step was defining three different goals in the world that are sufficiently distant from one another. The goals being distant would positively influence the perfomance of the assistance since there would then be a clearer distinction between the goals and thus the behaviors of the user can be interpreted in a more meaningful way. The three goals are shown in the figure below:

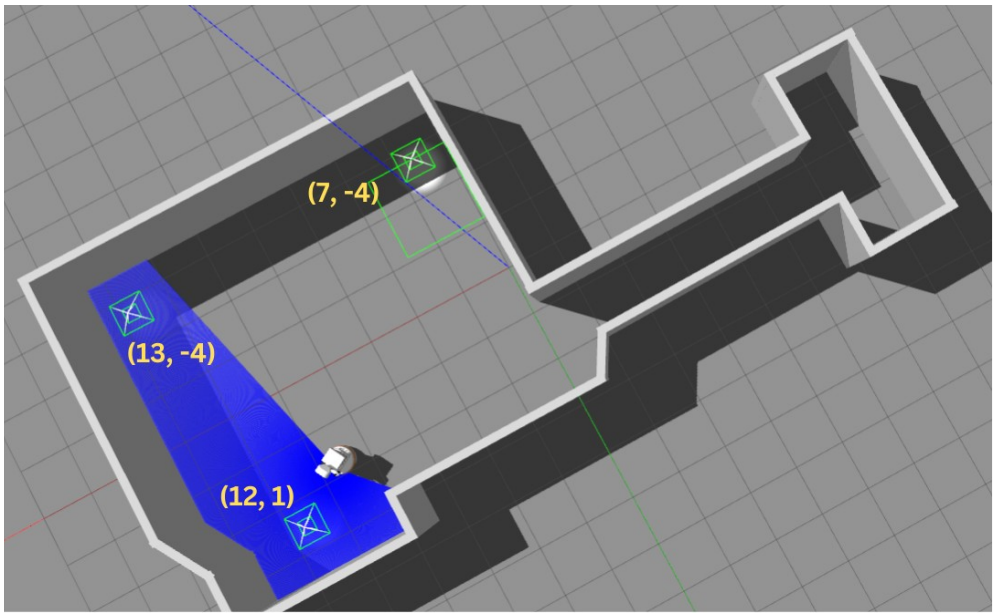

Fig. 30. TIAGo approaching one of the targets in Gazebo

The three goals with their positions are highlighted with the yellow text boxes. The robot would always start moving from the right side of the narrow hallway, in the small rectangular space, from a grid cell with the position of (0, 0). The goal positions (and later the constantly-changing center of masses) are defined with respect to this fixed frame and the robot has to transform these positions to its own reference frame to plan its motion towards them. Moreover, in the original setup of the simulated world, there are also obstacles located in the middle of the room which were removed for this section in order study the effects of the attractors more efficiently.

As it can be observed, the three goals are defined as distant as possible. The current assistance system only updates the probabilities in the case of receiving a non-zero

linear velocity from the user/keyboard. Rotational motions are ignored, as mentioned in previous chapters.

At the very beginning of the task, all the goals have the same probability of 33% since there is not enough information yet available to consider one single goal having a higher priority from the perspective of the user. As the program progresses, the probabilities get updated based on the series of linear velocity commands of the user. When the robot is close enough to one goal, there is a sudden significant increase in the probability dedicated to that goal. As it can be seen below, the robot has almost reached the goal (12, 1) and its probability is noticeably higher than that of other goals:

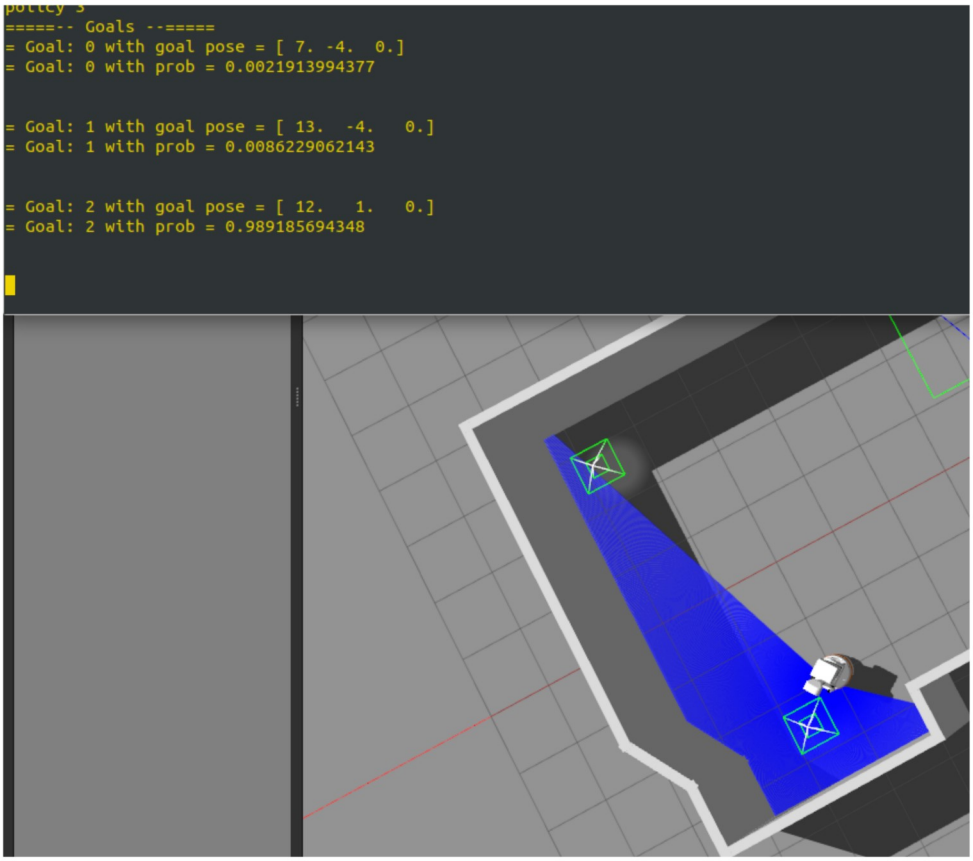

Fig. 31. The increase in the corresponding probability of the target towards which the robot is approaching

This is in fact an issue that was also addressed [23], as a shortcoming of global goal predictors, that the probability of the desired target would not increase just at the right time. In other words, the predictor would be able to *really* predict the actual user's goal when the robot has almost reached it.

In the figure below, the same scenario has reoccurred but this time for a different target. The robot has passed through the hallway and has almost reached the desired

target and just at this point, when the robot has reached this close, the predictor is able to properly predict the target. [13, -4, 0] is the target the robot is standing near to.

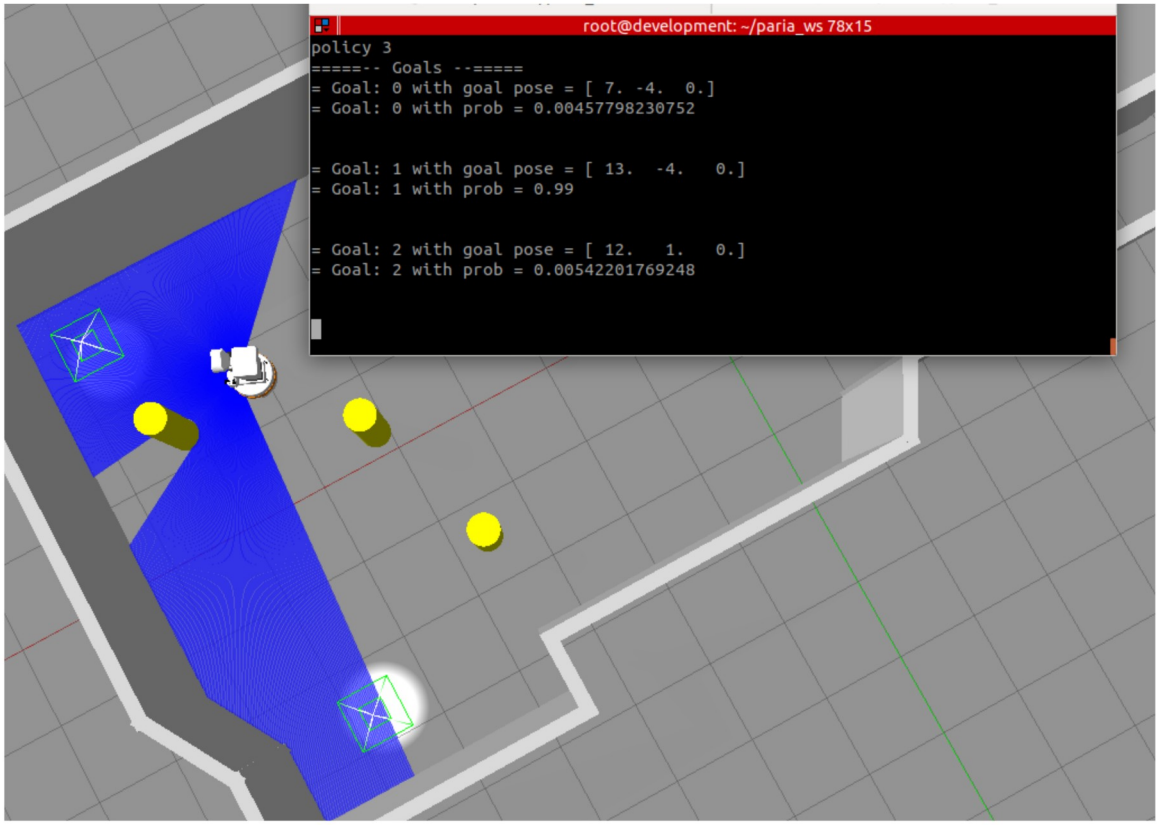

Fig. 32. Another case of the increase in the corresponding probability

Analyzing the project more in depth, we tested the system in the simulation and in each run the user had a different target to follow and had to be *consistent* in following it. In other words, if the user amid following one single goal changes his/her mind and decide to follow another target, the predicted probabilities would not be much reliable since they could experience a sudden increase or decrease and thus would not be much precise. Although, the predictor would eventually work properly in the sense that if the user starts approaching a different target, the corresponding probability would also increase but there's no guarantee that the increase in the newly chosen target would be satisfyingly rapid.

During the experiments, the robot would always start from the same position representing the position (0, 0), pass the narrow hallway and once it gets out, it would start following the targets based on the user's commands.

In the first experiment, the goal (12,1) was followed by the user. The robot is guided forward through a pure linear velocity command from the use, it gets out of the hallway and is initially attracted towards the other two goals since the region on the right of the robot is more populated with goals. The robot is then steered to its left by

the user who later keeping guiding it forward towards the (12,1). The robot keeps showing some resistance which is later mitigated when the user tries to approach the goal from a different position which becomes a successful attempt since the robot is now attracted more towards the actual chosen goal of the (12, 1). The video available at:
drive.google.com/file/d/1SS2CEz52OJf6BuZ-i2XFsVfEOSp8HMgU/view?usp=drive_link

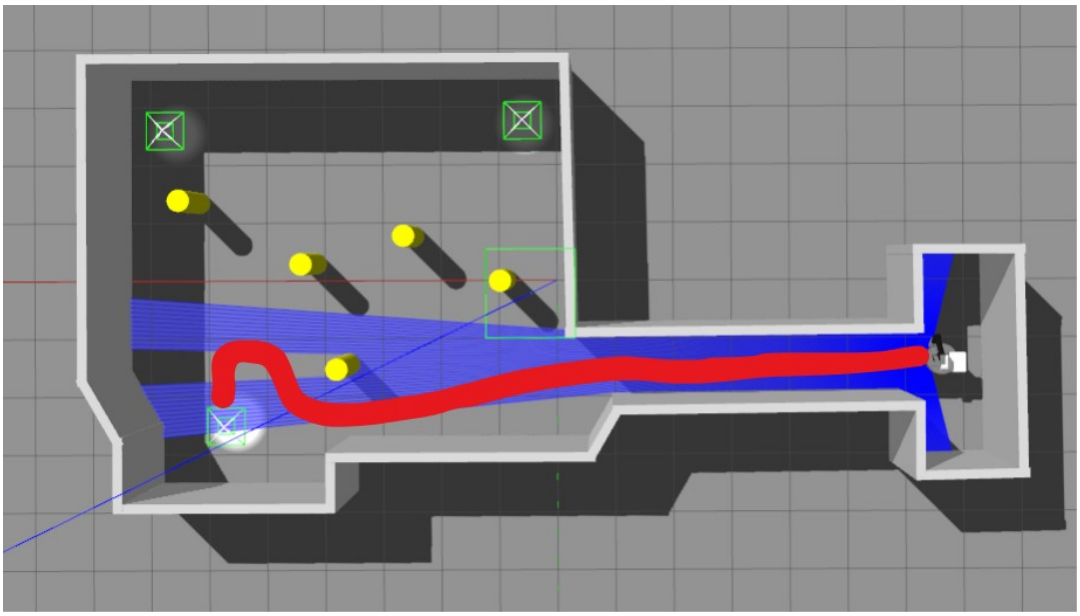

Fig. 33. The path to reach (12, 1)

The second experiment is dedicated to the user following goal at (13, -4). Similar to the previous case, the robot is instantly attracted towards its right after exiting from the narrow passage. The user continues to guide the robot forward and slightly steering it towards the chosen goal. The robot this time demonstrates some resistance and appears more attracted towards a center of mass that is situated somewhere between the (13, -4) and (12, 1). The user proceeds to move the robot forward towards (13, -4) when eventually the robot is successfully attracted to the chosen goal and does not show any resistance. The robot showing resistance would be the manifestation of a rapid rotation towards the opposite side of the where the user has been guiding it so far. The video available at:
drive.google.com/file/d/1VL2bCYdUVjoCX5CAeOsK4lV_yHCkhXqj/view?usp=drive_link

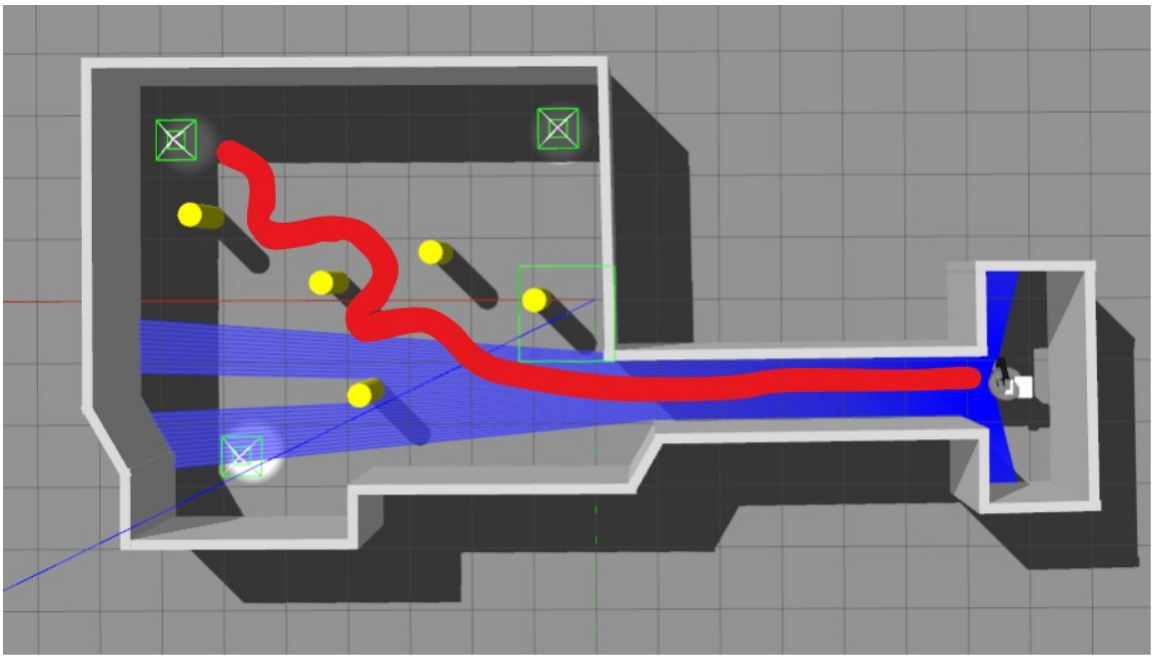

Fig. 34. The path to reach (13, -4)

Last by not least, the third goal or the one at (7, -4) appeared to be the most challenging for the robot to reach or be attracted to. Our assumption is that since the path towards this goal contains a high number of obstacles such as the walls and the yellow obstacle which the robot will face if it instantly steers right after exiting from the passage, the repulsive forces generated by these obstacles prevents the robot from being able to be attracted to a position beyond them. Moreover, if the user tries to guide the robot by first moving towards the center of the room and then guiding it to its right, this could potentially confuse the predictor since in the beginning it appears that the user is trying to guide the robot towards the goal at (13, -4).
In the video provided later in this paragraph, the user tries to approach the target from 2 different sides but in neither of them the robot manages to become attracted enough :*drive.google.com/file/d/1oSyrw2drCZXpIPpqiBJeYwZB07HSnb07/view?usp=drive_link*

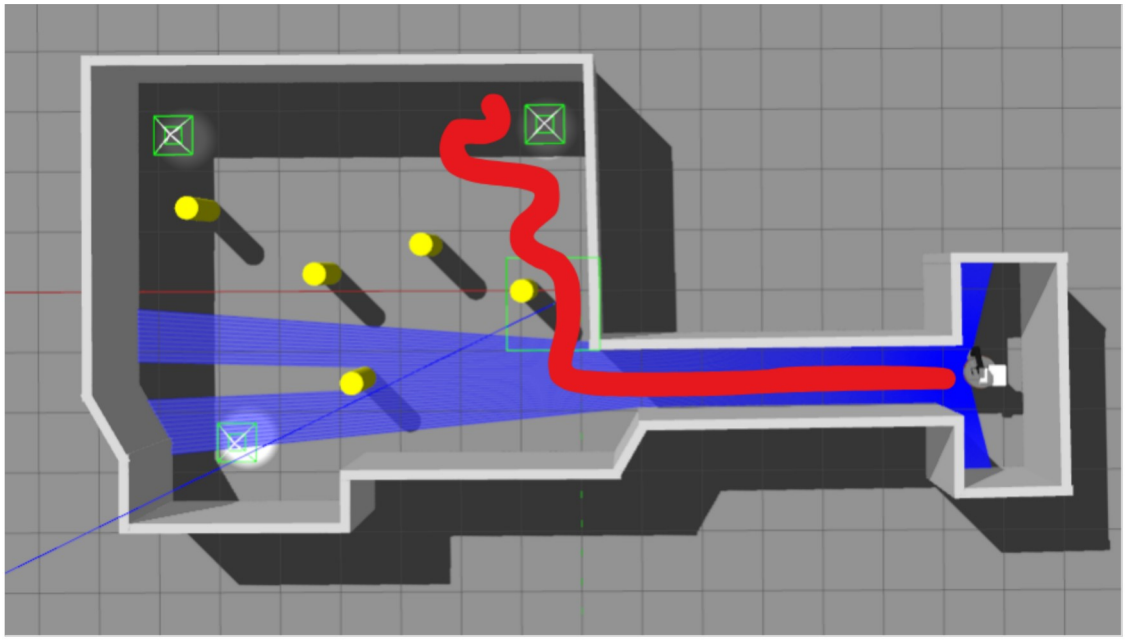

Fig. 35. The path to reach (7, -4), which fails in the end

7.4 Analytical results

For the case of the target (13, -4), we also gathered some analytical results.

To begin with, the information exposed in the rosbags retrived from the execution of the code highlight the changes in the probabilities of either of the goals. In the figure below, the probability values predicted for the target (13, -4) are depicted:

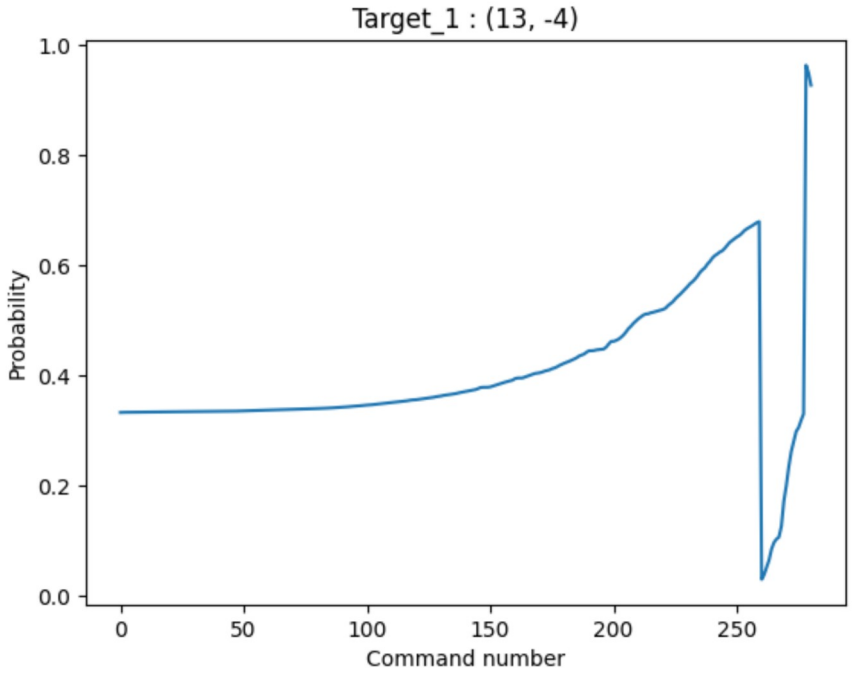

Fig. 36. Probability values for the target (13, -4)

The same data for the targets (7, -4) and (12, 1) are shown in figures 35 and 36 respectively:

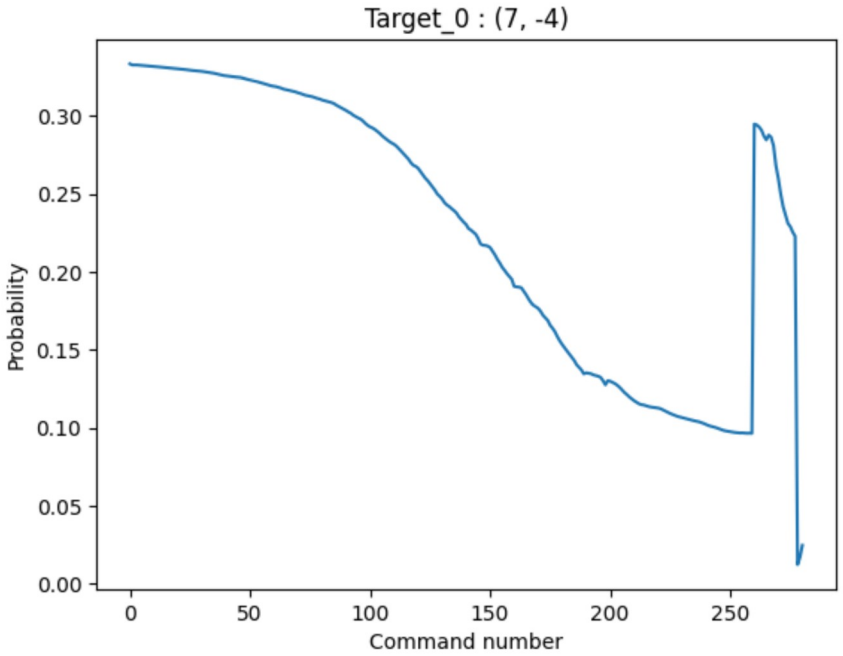

Fig. 37. Probability values for target (7, -4)

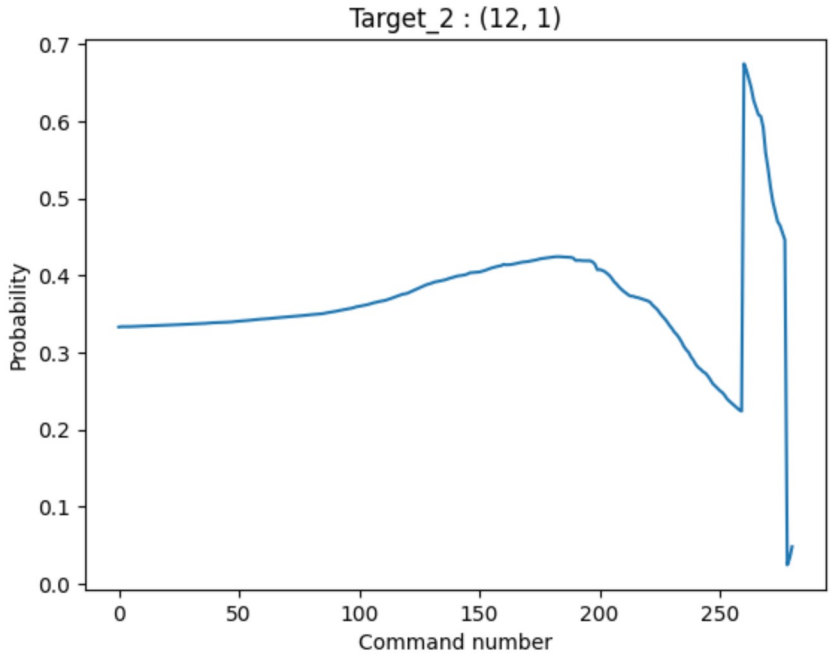

Fig. 38. Probability values for target (12, 1)

The target that was being followed by the user was (13, -4), looking at the graphs we see that the data of (7, -4) and (12, 1) are somehow the complementary of the data of (13, -4). In the beginning, all of the targets have a somewhat equal probability of

around 30%, when the probability for the (13, -4) starts to increase, the opposite trend is seen in the other two targets. Since in order for the robot to move towards (13, -4), the user should be guiding it towards the same side of the room in which (12, 1) is also situated, the probability values for (12, 1) are almost always higher than that of (7, -4). In the end, at the point in which there exists a sharp increase (above 90%) in the probability of (13, -4), there is also a sharp decrease in that of either of the other two targest.

Moreover, by visuallizing a scatter plot of the odometry information of the robot in the process of reaching (13, -4), the figure below was created:

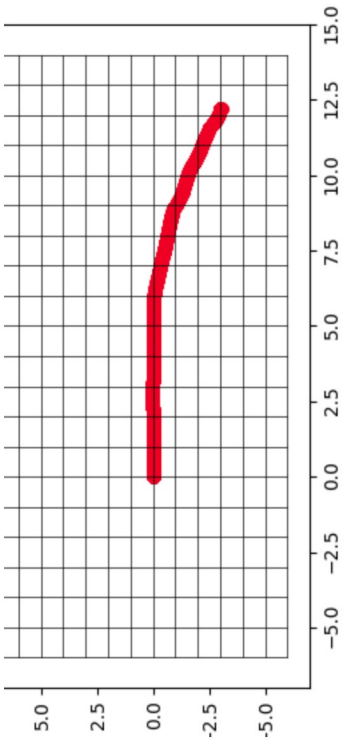

Fig. 39. The path from the starting pose of the robot to the target (13, -4)

The path to the target is not always quite smooth. In another experiment to reach the same target of (13, -4), the robot went on a different path starting from the same position as that in figure 37 and ending in the same position as that of the same figure

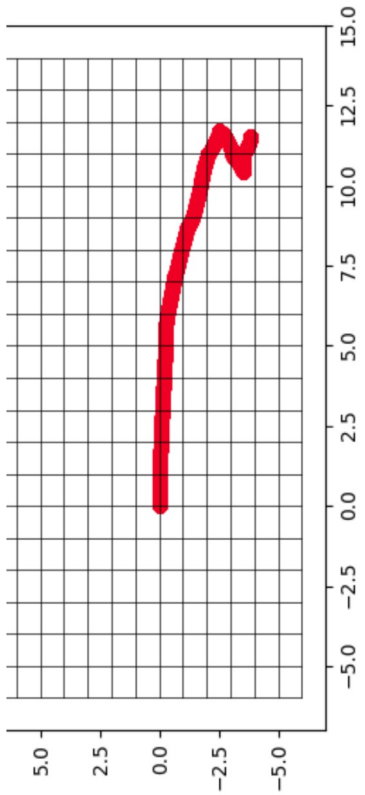

Fig. 40. A different path to reach the same target
as in figure 37

Relevant to the new experiment, we retrieved the probabilities for each target and plotted the data. There were more fluctuations in the outputs of the assistance system, since the user would have to try several times to guide the robot towards (13, -4) because the probability of (12, 1) was relatively high for a long period and so the robot would always try to steer towards (12, 1). That explains why in this experiment, after almost 370 user commands, the robot was finally attracted enough to the chosen target or put differently, its probablity got high enough (above 90%). While the same situation was reached much sooner in the previous experiment after only 260 commands.

Moreover, the path that robot traversed was more complex. The user had to send many angular commands towards the opposite direction in which the robot was trying to face (target (12, 1)) and hence a higher number of commands that were required to make the probability of the chosen target, reach as high as 90%.

Similar to the previous experiment, the probability plot of the chosen target is almost the complementary of that of the other two targets. This is observable especially at the times of sharp decreases/increases.

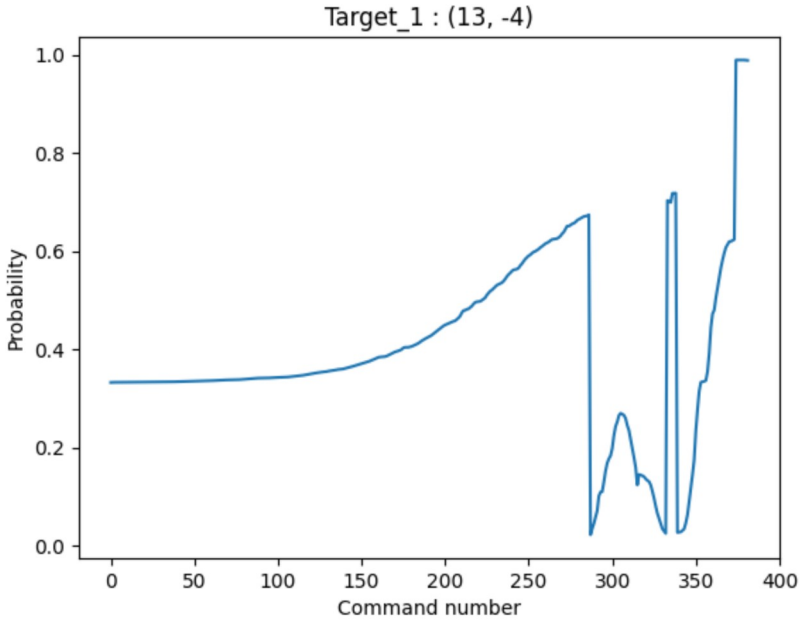

Fig. 41. Predicted Probabilities for target (13, -4)

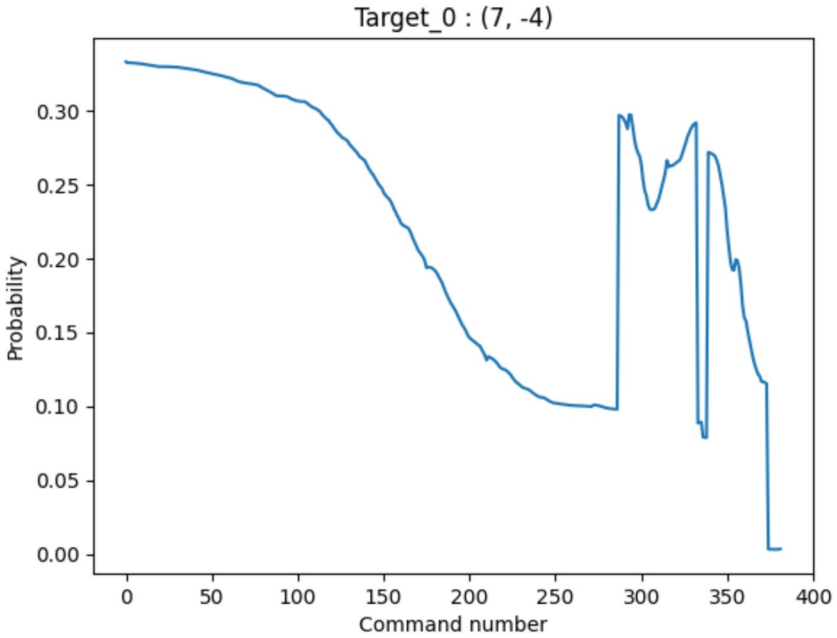

Fig. 42. Predicted probabilities for target (7, -4)

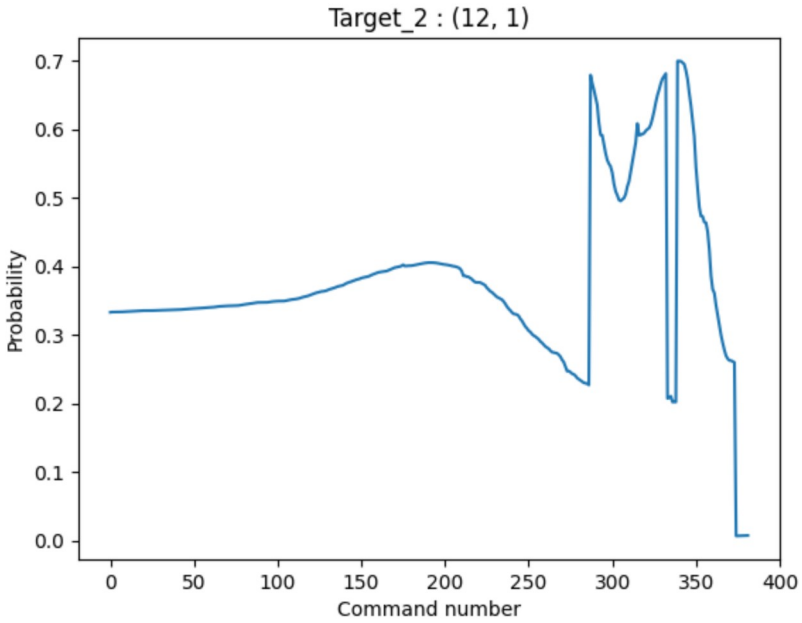

Fig. 43. Predicted Probabilities for target (12, 1)

7.5 Exepriments in the real world

Regarding the experiments with the real TIAGo, several measures had to be taken. First of all the chosen experiment environment had to be modified since the room contained tables which were located in the middle of the room and thus the laserscan of the robot would pass through to the other side, meaning that the robot would not consider the tables as obstacles. The solution was covering the area containing the tables, with a few panels in order to stop the laserscan.

Next was to create the map of the environment which was a process requiring the robot to fully scan the area. The map was later saved as a .pgm file and there was the possibility of choosing any map as the default one.

Fig. 44. The created map of the real-world environment

Fig. 45. The real-world environment where the three red dots represent the goals

Similar to the simulation, we first started with a pure teleoperation. This was possible through either a Logitech joystick or the keyboard. Next was to check the obstacle avoidance. It was observed that the robot is able to also avoid the dynamic obstacles that are put in its way. In other words, the repulsive field is generated also around the objects that are not already known and in the map. Although introducing new objects to the environment would later affect the localization of the robot as the robot would start getting lost or fail at doing mapping and localizing. There’s a video of the case

in which the robot is able to steer away from obstalces that are newly introduced and are not in the known map:
drive.google.com/file/d/17P6025roJ6DfTqHbkitlu-CJ9fAdSRuG/view?usp=drive_link

There are more videos available in which the similar final experiment that was done in the simulation was also carried out in the real-world environment.
Due to the complications of the real-world, the experiments could not be explored as much as they did in the simulations. Although, in the limited experiments that we managed to do with the real robot, the results were more or less similar to that of the simulation.
Perhaps one of the parameters that required much tuning was the weight given to the user velocity command and that computed by the APF. The safety was one of the important factors that had to be considered which was one of the elements that led to the user command velocities having a lower weight. Although this weight should not be so low so that it would make the user feel like that they are not really in control of the robot or the robot is not really doing what they are aiming for.
The robot would often end up behaving different in each run which was not always easy to guess the reason of since many times it would occur that by restarting the robot, the behavior would totally change, either in our favor or not.
The link to the videos:
drive.google.com/drive/folders/1bBjPHnJ-1JeZoxrPCmEhz9A7X_0zveJr

Conclusion and future work

In this thesis we discussed the shared control navigation of TIAGO based on the generation of artificial potential fields around the obstalces and the goals. Input velocity is coming from two sources: the user sending velocity commands using the keyboard and the local planner of the APF which manages the obstacle avoidance and helps the user in achieving the desired target by steering the robot towards the target.
The robot is equipped with a map of the environment and is therefore able to localize itself in it but it's unaware of the positioning of the goals and hence the need for an assistance system, predicting the most probable target of the navigation based on the overall behavior of the robot since the start of the interaction with the robot.
A limitation that was experienced several times during the experiments, was the performance of the assistance system in updating the probabilities. It's rather challenging to create the perfect sensitivity of the system to the user's behavior and it might happen that sometimes the probability of the intended goal not get updated either as soon as desired or as much as desired. Using a softmax function on the predicted probabilities could potentially mitigate this problem.
Another issue was the inconsistency of the TIAGo's localization in the real world. The robot would every now and then not manage to do the mapping between its sensory readings and the known map or experience some delay in updating its own pose. Moreover, it was observed that sending a sudden linear velocity of a high value could easily worsen this situation while sending a sequence of angular velocity commands could help the robot re-localize itself in the map. This was a rather big issue since the localization node and the relavant pose information that it publishes, are a crucial part of the work.
A possible way to extend this work could be by trying to implement the concept of *escape points* [11] which deals with the generation of a sequence of points in the cartesian space, on the surface of the convex obstacles. This was originally proposed as a solution for the fact that the local planner of the APF is prone to getting stuck in local minima. Having generated the sequence of escape points, the robot can safely modify its trajectory by tracing the escape points in order to pull itself out of a possible local minimum condition or to not fall into one in the first place.
Furthermore, it would be nice to assess the system using a qualitative questionare. Basically we would need several people teleoperating the robot in different modalities such as the pure teleoperation and shared-control teleoperation. The questionare could possibly contain questions such as whether the user was feeling in control, was the robot's behavior in line with the user's expectation, whether the user

had to put much effort in controlling the robot and whether their overal experience was positive with either of the modalities.

Bibliography

[1] pal-robotics.com/robots/tiago/

[2] “Introduction to AI Robotics.” MIT Press, https://mitpress.mit.edu/9780262038485/introduction-to-ai-robotics/. Accessed 24 June 2024.

[3] nasa.gov/image-article/astronaut-john-young-collecting-lunar-samples

[4] leorover.tech/post/what-was-the-worlds-first-mobile-intelligent-robot

[5] Beraldo, Gloria, et al. “Brain-Driven Telepresence Robots: A Fusion of User’s Commands with Robot’s Intelligence.” AIxIA 2020 – Advances in Artificial Intelligence, edited by Matteo Baldoni and Stefania Bandini, Springer International Publishing, 2021, pp. 235–48. Springer Link, https://doi.org/10.1007/978-3-030-77091-4_15

[6] Loganathan, Anbalagan, and Nur Syazreen Ahmad. “A Systematic Review on Recent Advances in Autonomous Mobile Robot Navigation.” Engineering Science and Technology, an International Journal, vol. 40, Apr. 2023, p. 101343. DOI.org (Crossref), https://doi.org/10.1016/j.jestch.2023.101343

[7] Huang, L. “Velocity Planning for a Mobile Robot to Track a Moving Target — a Potential Field Approach.” Robotics and Autonomous Systems, vol. 57, no. 1, Jan. 2009, pp. 55–63. DOI.org (Crossref), https://doi.org/10.1016/j.robot.2008.02.005

[8] Khatib, Oussama. “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots.” The International Journal of Robotics Research, vol. 5, no. 1, Mar. 1986, pp. 90–98. DOI.org (Crossref), https://doi.org/10.1177/027836498600500106

[9] Teli, Tawseef Ahmed, and M. Arif Wani. “A Fuzzy Based Local Minima Avoidance Path Planning in Autonomous Robots.” International Journal of Information Technology, vol. 13, no. 1, Feb. 2021, pp. 33–40. DOI.org (Crossref), https://doi.org/10.1007/s41870-020-00547-0

[10] Hoshino, S., and K. Maki. “Safe and Efficient Motion Planning of Multiple Mobile Robots Based on Artificial Potential for Human Behavior and Robot Congestion.” Advanced Robotics, vol. 29, no. 17, Sept. 2015, pp. 1095–109. DOI.org (Crossref), https://doi.org/10.1080/01691864.2015.1033461

[11] Gottardi, Alberto, et al. “Shared Control in Robot Teleoperation With Improved Potential Fields.” IEEE Transactions on Human-Machine Systems, vol. 52, no. 3, June 2022, pp. 410–22. DOI.org (Crossref), https://doi.org/10.1109/THMS.2022.3155716.

[12] Beraldo, Gloria, et al. “Shared Autonomy for Telepresence Robots Based on People-Aware Navigation.” Intelligent Autonomous Systems 16, edited by Marcelo H. Ang Jr et al., Springer International Publishing, 2022, pp. 109–22. Springer Link, https://doi.org/10.1007/978-3-030-95892-3_9

[13] Chen, Yong-bo, et al. “UAV Path Planning Using Artificial Potential Field Method Updated by

Optimal Control Theory.” International Journal of Systems Science, vol. 47, no. 6, Apr. 2016, pp. 1407–20. DOI.org (Crossref), https://doi.org/10.1080/00207721.2014.929191.

[14] García, Alberto, et al. “Portable Multi-Hypothesis Monte Carlo Localization for Mobile Robots.” 2023 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2023, pp. 1933–39. DOI.org (Crossref), https://doi.org/10.1109/ICRA48891.2023.10160957.

[15] Tonin, Luca, et al. “Learning to Control a BMI-Driven Wheelchair for People with Severe Tetraplegia.” iScience, vol. 25, no. 12, Dec. 2022, p. 105418. DOI.org (Crossref), https://doi.org/10.1016/j.isci.2022.105418.

[16] wiki.ros.org/rviz/UserGuide

[17] docs.pal-robotics.com/ari/sdk/23.1.11/development/deploy-code.html

[18] github.com/braingear-wheelchair

[19] github.com/shared-control

[20] Lin, Rui, et al. “Ultra-Wide-Band-Based Adaptive Monte Carlo Localization for Kidnap Recovery of Mobile Robot.” International Journal of Advanced Robotic Systems, vol. 20, no. 2, Mar. 2023, p. 172988062311639. DOI.org (Crossref), https://doi.org/10.1177/17298806231163950.

[21] wiki.ros.org/turtlesim

[22] Javdani, Shervin, et al. “Shared Autonomy via Hindsight Optimization for Teleoperation and Teaming.” The International Journal of Robotics Research, vol. 37, no. 7, June 2018, pp. 717–42. DOI.org (Crossref), https://doi.org/10.1177/0278364918776060

List of Figures

13 “(a) A typical circular repulsive field (b) behvaior potential field generated for a human with a slow pace (c) behavior potential field generated for a human with a faster pace”
doi.org/10.1080/01691864.2015.1033461

14 “Adjacent repulsive fields”
doi.org/10.1080/01691864.2015.1033461

15 “The robot will be attracted towards a newly defined virtual target while it’s being repelled by the exterme points on the obstacle as a way of reassuring that it won’t fall into the local minimum again”
doi.org/10.1007/s41870-020-00547-0

16 “Path generated by APF”
doi.org/10.1007/s41870-020-00547-0

17 “Path generated by the fuzzy block”
doi.org/10.1007/s41870-020-00547-0

18 “Typical ANN with two hidden layers”
doi.org/10.1016/j.jestch.2023.101343

19 “Dense particles”
Gazebo

20 “Rather widespread particles”
Gazebo

21 “Acceptable localization”
Gazebo

22 “Fast linear motion of the robot might sometimes make the robot disoriented with respect to the known map”
Gazebo

23 “Problem of figure 21 gets resolved by a sequence of angular velocity commands ”
Gazebo

24 “Proximity grid around TIAGo”
Gazebo

25 “Theta is the angle between the heading direction of the robot and the vector representing the euclidean distance between the target and the goal”
Gazebo

26 “TIAGo”
pal-robotics.com/robots/tiago/

27 “The simulated world on Gazebo. The yellow cylindars are the obstacles

and the targets are highlighted using green squares”
Gazebo

28 “TIAGo not being able to exit from the narrow hallway due to a high strength given to the attractors”
Gazebo

29 “TIAGo successfully exiting the narrow hallway after a reduction in the strength of the attractors ”
Gazebo

30 “TIAGo approaching one of the targets in Gazebo ”
Gazebo

31 “The increase in the corresponding probability of the target towards which the robot is approaching”
Gazebo

32 “Another case of the increase in the corresponding probability”
Gazebo

33 “The path to reach (12, 1)”
Gazebo

34 “The path to reach (13, -4)”
Gazebo

35 “The path to reach (7, -4), which fails in the end”
Gazebo

36 “Probability values for the target (13, -4)”
made with Matplotlib from the recorded rosbags

37 “Probability values for target (7, -4)”
made with Matplotlib from the recorded rosbags

38 “Probability values for target (12, 1)”
made with Matplotlib from the recorded rosbags

39 “The path from the starting pose of the robot to the target (13, -4)”
made with Matplotlib from the recorded rosbags

40 “A different path to reach the same target as in figure 37”
made with Matplotlib from the recorded rosbags

41 “Predicted probabilities for target (13, -4)”
made with Matplotlib from the recorded rosbags

42 “Predicted probabilities for target (7, -4)”
made with Matplotlib from the recorded rosbags