Koordinirano planiranje gibanja i upravljanje mobilnim robotima za zajednički prijevoz velikog tereta

Paro, Borna

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 72

COORDINATED MOTION PLANNING AND CONTROL OF MOBILE ROBOTS FOR JOINT TRANSPORTATION OF LARGE OBJECTS

Borna Paro

Zagreb, June 2024

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 72

COORDINATED MOTION PLANNING AND CONTROL OF MOBILE ROBOTS FOR JOINT TRANSPORTATION OF LARGE OBJECTS

Borna Paro

Zagreb, June 2024

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zagreb, 04 March 2024

MASTER THESIS ASSIGNMENT No. 72

Student:	**Borna Paro (0069088363)**
Study:	Information and Communication Technology
Profile:	Control Systems and Robotics
Mentor:	prof. Ivan Marković

Title: **Coordinated motion planning and control of mobile robots for joint transportation of large objects**

Description:

The aim of the thesis is to implement a method for coordinated motion planning and control of two autonomous mobile robots for the joint transport of an object that is too large for a single robot. The method will be implemented within the Robot Operating System (ROS) environment and validated in simulation scenarios. The simulation will include two robots that first arrive independently at the location of the object and then jointly with the object form a new kinematic chain. Thence, an optimal trajectory is planned for the entire kinematic chain, which ensures the transport of the large object to the desired target location. After that, the implemented algorithm sends control commands to each robot separately, which achieves the coordinated execution of the planned path. The proposed approach can be widely applied in areas such as logistics and industrial robotics, contributing to the optimization of load transfer processes and improving the efficiency of autonomous robots in manufacturing environments.

Submission date: 28 June 2024

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 72

Pristupnik:	**Borna Paro (0069088363)**
Studij:	Informacijska i komunikacijska tehnologija
Profil:	Automatika i robotika
Mentor:	prof. dr. sc. Ivan Marković

Zadatak: **Koordinirano planiranje gibanja i upravljanje mobilnim robotima za zajednički prijevoz velikog tereta**

Opis zadatka:

Cilj diplomskoga rada je implementirati metodu za koordinirano planiranje gibanja i upravljanje dvama autonomnim mobilnim robotima za zajednički prijevoz tereta koji je dimenzijama prevelik za jednog robota. Metoda će biti implementirana unutar Robot Operating System (ROS) okruženja te validirana u simulacijskim scenarijima. Simulacija će uključivati dva robota koji prvo nezavisno dolaze do lokacije tereta te zajedno s teretom čine novi kinematički lanac. Zatim se planira optimalna putanja za cijeli kinematički lanac koja osigurava prijevoz tereta na željenu ciljnu lokaciju. Nakon toga, implementirani algoritam šalje upravljačke naredbe svakom robotu zasebno, čime se postiže koordinirano izvršavanje planirane putanje. Predloženi pristup može imati široku primjenu u područjima kao što su logistika i industrijska robotika, pridonoseći optimizaciji procesa prijenosa tereta i poboljšanju učinkovitosti rada autonomnih robota u proizvodnim okruženjima.

Rok za predaju rada: 28. lipnja 2024.

Zahvaljujem se ovim putem mentoru Ivanu Markoviću te asistentima Luki Petroviću i Jeleni Gregorić na responzivnosti, odvojenom vremenu, raznim savjetima i pomoći koji su bili od velikog značaja u izradi ovog rada.

Zahvaljujem se Bogu, obitelji, svim prijateljima, kolegama, profesorima i asistentima koji su mi na bilo koji način pomogli, olakšali i uljepšali studiranje.

Za kraj, posebno se zahvaljujem svojim roditeljima, ocu Žoržu i majci Marini, koji su mi bili podrška u svemu i omogučili sve u životu, pa tako i bezbrižno studiranje u drugom gradu. Hvala Vam.

Contents

1 Introduction

Robotics is an interdisciplinary branch of computer science and engineering that includes designing, creating and programming robots. Robotics creates machines that can replace and replicate human actions. Robots can be used in various situations, for example, in dangerous environments for humans such as finding and deactivating explosive devices, going into space, working at high temperatures, working in polluted and radioactive areas, etc. [1]. Although robots today are mostly used in extreme conditions, such as mentioned above, many robots also perform tasks that humans find tedious, repetitive or boring. Robots can require human control or they can be completely autonomous, which is of particular interest because it requires almost no human foreknowledge, while a robot does everything that is required of it by itself.

A mobile robot is an automated machine that is capable of locomotion [2]. Mobile robots have the capability to move around in their environment and are not fixed to one physical location. Mobile robots can be *autonomous*, meaning they are capable of navigating an uncontrolled environment without the need for physical or electro-mechanical guidance devices [3]. By contrast, industrial robots are usually more-or-less stationary, consisting of a jointed arm (multi-linked manipulator) and gripper assembly (or end effector), attached to a fixed surface. Mobile robots have become more commonplace in commercial and industrial settings. Hospitals have been using autonomous mobile robots to move materials for many years. Warehouses have installed mobile robotic systems to efficiently move materials from stocking shelves to order fulfillment zones, while robot cleaners and robot lawn mowers are becoming more and more popular. Mobile robots are also found in industrial, military and security settings.

Transport of large cargo is a frequent task that needs to be done on working sites to transport it from the warehouse to the place where that cargo is needed. Instead of

people transporting the cargo manually with trucks or vans from one point to another, a more efficient approach would be to have robots transporting it, while people could focus on creative tasks. Moreover, if there are multiple robots at disposal that can transport cargo, but that cargo is too large or too heavy for a single robot, instead of purchasing an additional robot that can carry alone all that weight and size, it would be more cost-effective to use two less expensive and more available robots to transport it.

This thesis presents a solution to a problem where two robots have to carry a large load on themselves, which is too big for a single robot, and with it find an optimal path from the starting pose to the goal pose, bypassing obstacles and maneuvering through narrow corridors. All developed methods are implemented using Robot Operating System and programming languages Python and C++. They are simulated and visualized using Gazebo and RViz as well as tested in a real world environment.

The thesis is organized as follows. In Chapter 2, the problem that is trying to be solved is presented, as well as prior researchers' work tackling similar problems. In Chapter 3, technologies and tools used throughout the thesis are described. In Chapter 4, a detailed explanation of the implemented algorithms and system architecture and workflow is presented, as well as the initial attempts and problems encountered with them. Results of the experiments, both in simulation and real world environment, are shown in Chapter 5. In Chapter 6, we present the successful deployment of the implemented methods and discuss differences between simulation and real world results, as well as potential solutions to the mentioned problems.

2 Problem description and prior work

The problem we are trying to solve is having two independent robots and a large object of arbitrary shape, mounted on top of them, that we want to transport. We want to find a collision free path and calculate the velocities which guide the structure towards the goal position without collisions with the environment. We want to model this structure of two robots and a large object as a single entity, calculate the path and velocities for it, and then transform those velocities to velocities for each robot that achieve the desired motion of the whole structure and follow the calculated path.

The authors in [4] presented a method for multi-robot formation control in dynamic environments that avoids collisions with static and moving obstacles. In [5], a method that allows two wheeled mobile robots (leader and follower) to navigate through known environments while cooperatively carrying an object, was presented. A motion planning method of multiple mobile robots for cooperative transportation of a large object in a three-dimensional environment was presented in [6]. A distributed leader–helper architecture for teams of two autonomous mobile robots that jointly transport large payloads while avoiding collisions with obstacles (either static or dynamic) was introduced in [7].

Most of the methods use some kind of a leader-follower configuration for solving the problem of multi-robot cargo carrying tasks, but none of them uses kinematic chain to model the structure and its motion. The main contribution of this thesis would be motion and path planning algorithms for anyshape kinematic chain that consists of two robots and a large cargo that they are carrying. The robots would be independent of one another, i.e. the motion commands for one robot would not depend on the motion of the other, they would only depend on the poses that kinematic chain structure needs to

achieve through the path so the motion from start to the goal would be collision free. In that way, each robot node would have fewer parameters to calculate so the computations would be faster and they can work independently of one another, without knowing that the other robot exists.

3 Technologies and tools used

In this chapter, technologies and tools that were used in the creation of this thesis will be described.

3.1 Robot Operating System (ROS)

Robot Operating System (abbreviated ROS) is a collection of open source middlewares that serve for developing software for controlling robots [8]. Although it has an operating system in its name because of the functionality it provides, such as: hardware abstraction, control devices at a low level, implementation of common functionalities, transmission of messages between processes and package management etc., it is not really a stand-alone operating system, but must be installed on an operating system such as GNU/Linux.

Computation graph model

ROS processes are represented as nodes in a graph structure, connected by edges called topics [9]. ROS nodes can pass messages to one another through topics, make service calls to other nodes, provide a service for other nodes, set/retrieve shared data from a communal database called the parameter server. A process called the ROS Master [9] makes all of this possible by registering nodes to itself, setting up node-to-node communication for topics, and controlling parameter server updates.

Example of ROS computational graph is shown in the figure 3.1. .

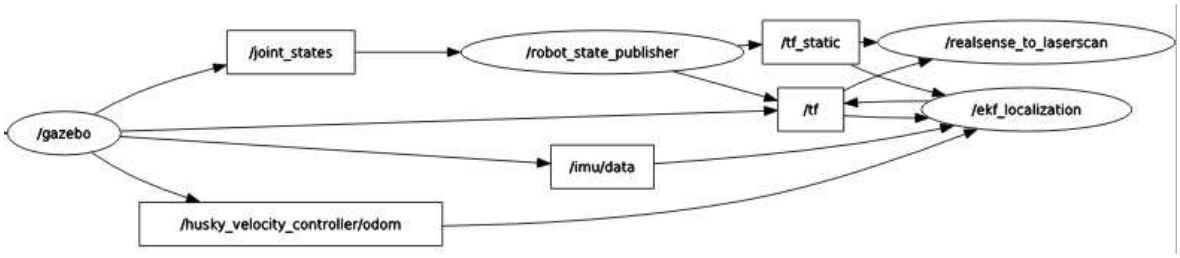


Figure 3.1: ROS computational graph example

Nodes

A node represents one process running the ROS graph [9]. Every node has a name, which it registers with the ROS master before it can take any other actions. Multiple nodes with different names can exist under different namespaces, or a node can be defined as anonymous, in which case it will randomly generate an additional identifier to add to its given name.

In practice, a robotic system consists of several nodes, each of which performs its own task. For example, one node sends velocity commands to wheels, the other serves for calculating odometry, the third visualizes the trajectory etc.

Messages

Nodes communicate with each other by passing messages [10]. Message is a data structure that can be simple, for example an integer, but it can also be complex so that it contains arrays, instances of other messages etc.

Topics

Topics are named buses over which nodes send and receive messages [11]. Topic names must be unique within their namespace as well. To send messages to a topic, a node must publish to said topic, while to receive messages it must subscribe. The publish/subscribe model is anonymous: no node knows which nodes are sending or receiving on a topic, only that it is sending/receiving on that topic. The types of messages passed on a topic vary widely and can be user-defined. The content of these messages can be sensor data, motor control commands, state information, actuator commands, or anything else. Unlike services, publishing and subscribing to the topic can be aborted.

Services

Service consists of a server that waits for a request and a client that sends a request to the server [12]. Service is mainly a short, simple action that is done quickly and is not repetitive and happens rarely. A service request cannot be aborted, unlike to topics and actions.

Actions

Action is a combination of topics and services and is used for works that require a longer time to complete, for example - mobile robot path following. The client and server communicate with three kinds of messages: *goal*, *feedback* and *result* [13]. Action starts with the client making a request and sending goal message to the action server, action server accepts it, and until the action is finished, sends to the client information about the execution as feedback messages. When action completes the goal objective, server sends result information via result message to the client.

3.2 Gazebo simulator

Gazebo is an open source 2D/3D simulator for simulating robots and their environment. It has an integrated *ODE* system for simulating physics over objects, *OpenGL* for rendering and support for simulating various sensors and actuators [14]. It is a very popular simulator because of its high performance and because of the graphical interface through which you can easily model the world with various obstacles and you can easily model robots with various configurations and sensors that will be used.

3.3 RViz

ROS visualization or abbreviated RViz is a 3D graphical interface that serves to visualize various data that occur within the ROS system [15]. With the help of various plugins, various information can be displayed, such as a map, transformations, information about the robot model, sensor information, markers, etc.

3.4 Husky robot

For simulation and real world testing, Husky robot created by Clearpath Robotics [16] was used. At first Pioneer 3-DX, created by Adept [17] was chosen, but problems occurred using ROS navigation stack with it, specifically *move_base* package. Husky robot was chosen because other students worked with it and everything worked fine, also because we have it in our laboratory so the real experiments could be tried with it. Husky robot can be seen in the following figure (Figure 3.2):

Figure 3.2: Clearpath Husky robot

.

3.5 Intel RealSense Depth Camera D435i

For real world testing, instead of laser scanner, we opted for camera [18] because of its easy usage. Everything that needs to be done is connect it to the laptop and install drivers for it and then it is *plug-and-play*. Since nodes for localization, SLAM and obstacle de-

tection use laser information, pointcloud received from the camera first needs to be converted to the laser scan. Camera used in real world testing is shown in the following figure (Figure 3.3):

Figure 3.3: RealSense Depth Camera D435i

3.6 Docker

Docker is a set of *platform as a service* products that use operating system virtualization to deliver software in packages called containers [19]. Docker is like a lightweight virtual machine with less graphical interface.

For experimenting with two real robots, two laptops are needed. The other laptop that we used had Ubuntu 22.04 installed, since ROS noetic runs only on Ubuntu 20.04, instead of installing another operating system, we opted for using Docker because it required much less time to set up.

3.7 NetworkX

NetworkX is a Python package used for creating graphs, searching through them, manipulating them, etc. [20]. Package was used in our implementation of a global planner for creating and searching through graph data structure.

4 The proposed method

In this chapter, the global path planning method of the arbitrary shape structure will be presented. The motion planning method used for the purposes of this work will also be described. Finally, it will be briefly described how the system works in its entirety.

4.1 Global path planning

In robotics, many path planning and motion execution algorithms, for the sake of efficiency, approximate the robot footprint with a circle [21]. Using that approximation, obstacles can be inflated by the robot's radius and then robot can be displayed as single dot for which path planning is a simple graph search. This assumption potentially limits the mobility of non-circular robots especially in confined environments such as corridors, narrow passages or even in dynamic environments. In our setup, with two robots and long cargo, approximating the structure with circle would make a footprint unnecessarily big which would result in the inability to find a path in the environment where the structure could pass through with little bit of maneuvering.

The global path planner's task is to quickly generate a path from the current robot's pose to the goal pose for a known map. It also serves as proof if the robot can even reach that pose, if the global path is empty - it means that robot cannot arrive to that goal pose, e.g. because it cannot bypass static obstacles, it would be in collision with obstacles at that pose, that pose is already occupied etc. After calculating global path, it is forwarded to the local planner who follows it and, if necessary, with the help of sensors, detects obstacles that are not known in advance in the static map and bypasses them. ROS global planners take robot's footprint and then approximate it with a circle [22], which is, as stated before, inadequate for this problem. Therefore, in this thesis we present our implementation of global path planning algorithm for arbitrary shape

differential drive robot based on the paper [23].

Global planner works as follows: planner receives static map from the map server and robot's footprint from the costmap. Map is divided into cells of CELL_SIZE x CELL_SIZE dimensions, where CELL_SIZE is a parameter given to the node which depends on the map size, the smaller the map the smaller the CELL_SIZE needs to be so the planner wouldn't give false negatives for the path that actually exists. First, admissible orientations need to be calculated for every cell. Admissible orientations are orientations with which robot isn't in collision with any obstacle while positioned in the middle of that cell. Algorithm loops through every cell, if the cell by itself is occupied, then there isn't any admissible orientation for that cell and algorithm moves forward, if it isn't occupied by itself, algorithm loops through predefined list of orientations that a structure can achieve, and for every orientation - it checks if the footprint is in collision with any obstacle, if it isn't - then that orientation is added to the list of admissible orientations for that cell.

Collision checking part of the algorithm is visualized in a figure 4.1. Gray lines represent cells. Structure's footprint is placed in a center of a cell and rotated by orientation θ. Ray is traced from every point of a footprint to the center of a cell, if ray passes through an occupied area for any point - function returns that structure is in collision with that orientation and that orientation isn't added to the admissible ones. If ray for every point of footprint doesn't pass through the occupied area, function returns that structure is not in collision for that orientation. In the figure, footprint on the left is in collision for $\theta = 135°$, while footprint on the right isn't.

After admissible cell orientations are calculated for every cell, graph needs to be created so the graph search algorithm can be run through it. Every cell is a node in a graph, for every node, algorithm takes into account its eight neighbours. Since structure has differential drive, it cannot move laterally, therefore two nodes will be connected, i.e. have edge between them, only if current and neighbouring node both have in their admissible orientations orientation that is needed for robot in current cell to traverse to that neighbouring cell. Visualization and further explanation are shown in figure 4.2.

For searching the graph, Dijkstra algorithm [24] is used where edge weight is calcu-

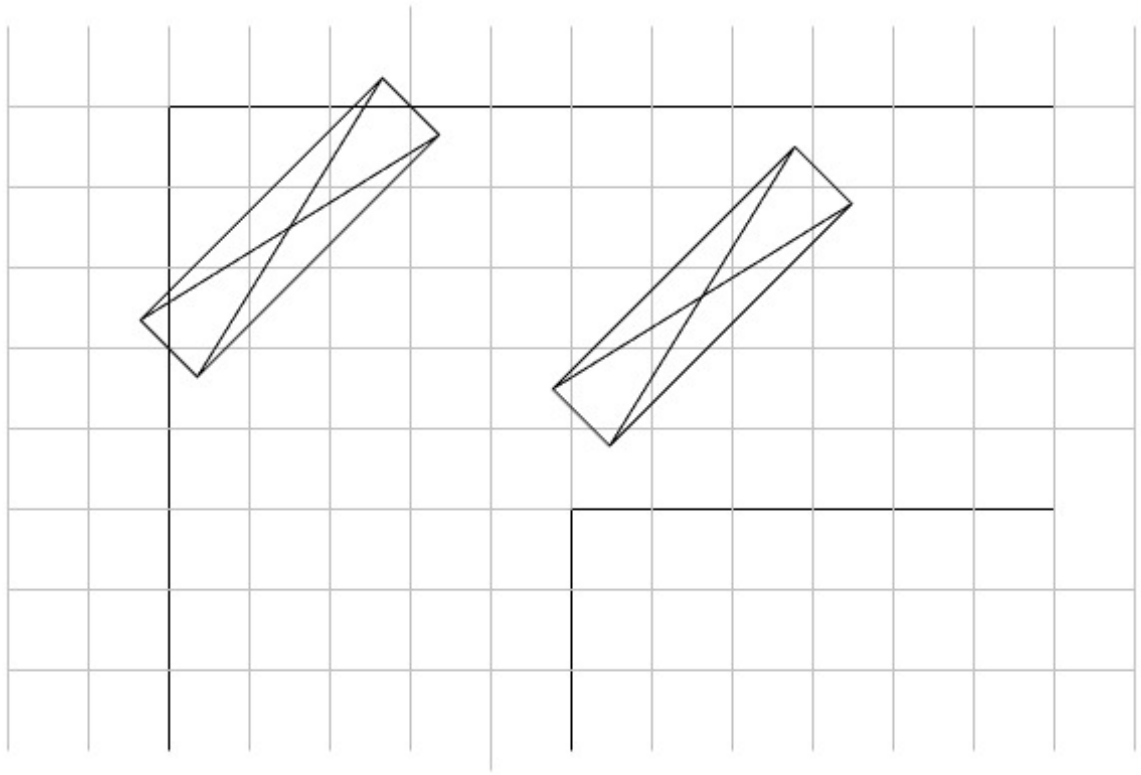


Figure 4.1: Collision checking visualization

lated by following formula which is similar to one used in a paper [23]:

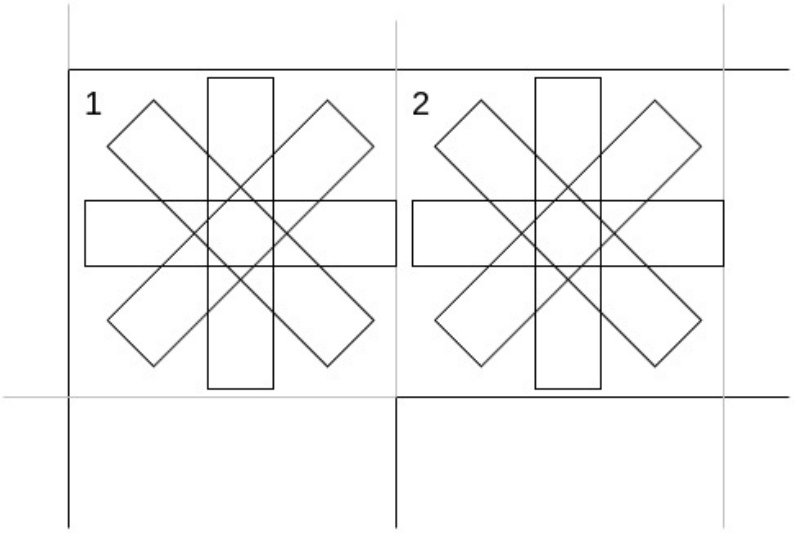
$$w = M - |\Theta_c \cap \Theta_n| + 100 \cdot [[\theta_d \in \Theta_{diag}]]$$

where: M is a number of all orientations that robot can potentially achieve, Θ_c is set of admissible orientations of the current cell, Θ_n is set of admissible orientations of the neighbouring cell, θ_d is needed orientation that robot in current cell needs to have so it could traverse to the neighbouring cell, Θ_{diag} are orientations with which robot moves diagonally (e.g. 45 deg, 135 deg, etc.).

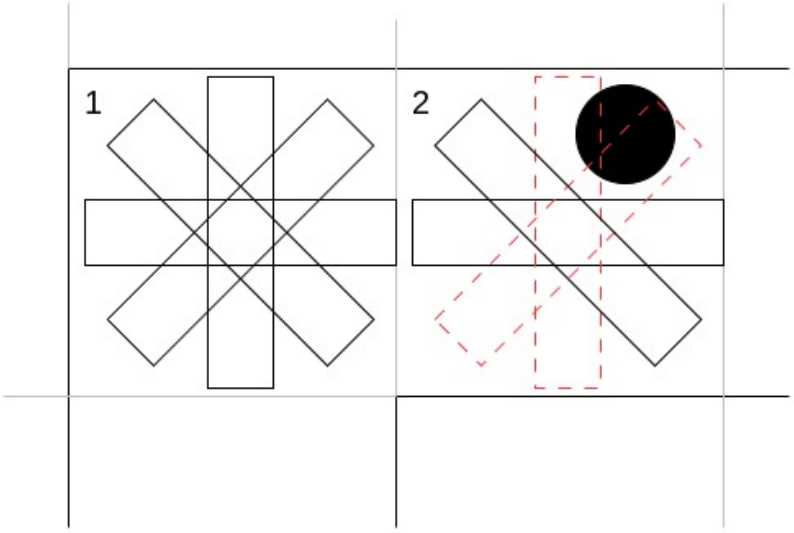
Using this weight function, algorithm will choose a path with cells that have more orientations in common, also it will rather choose non diagonal movement which ensures smoother path, therefore smoother movement.

Example of a global path with the footprint and its orientation shown for each cell of a path can be seen in the figure 4.3.

.



(a) Example without obstacle



(b) Example with obstacle

Figure 4.2: Structure's admissible orientations are shown for the cells one and two. For structure to move from cell one to cell two, which is its direct right neighbour, it needs to move by having an orientation of $\theta = 0°$. Algorithm checks if the cell one in its admissible orientations has orientation of 0°, which means that structure, if needed, can rotate in place so that it aquires desired orientation. If that's true, algorithm checks if the cell two has $\theta = 0°$ in its admissible orientations, which would mean that structure can arrive to that cell with that orientation without colliding with obstacle. If that is also true, then edge exists between those two nodes (cells). Contrary, in subfigure 4.2b, structure would be in collision with obstacle for orientation $\theta = 0°$, and in that case, it cannot move from cell one to cell two so those two nodes aren't connected. Note: for simplicity, easier readability and understanding, only a few of orientations are shown, and cell size is much larger

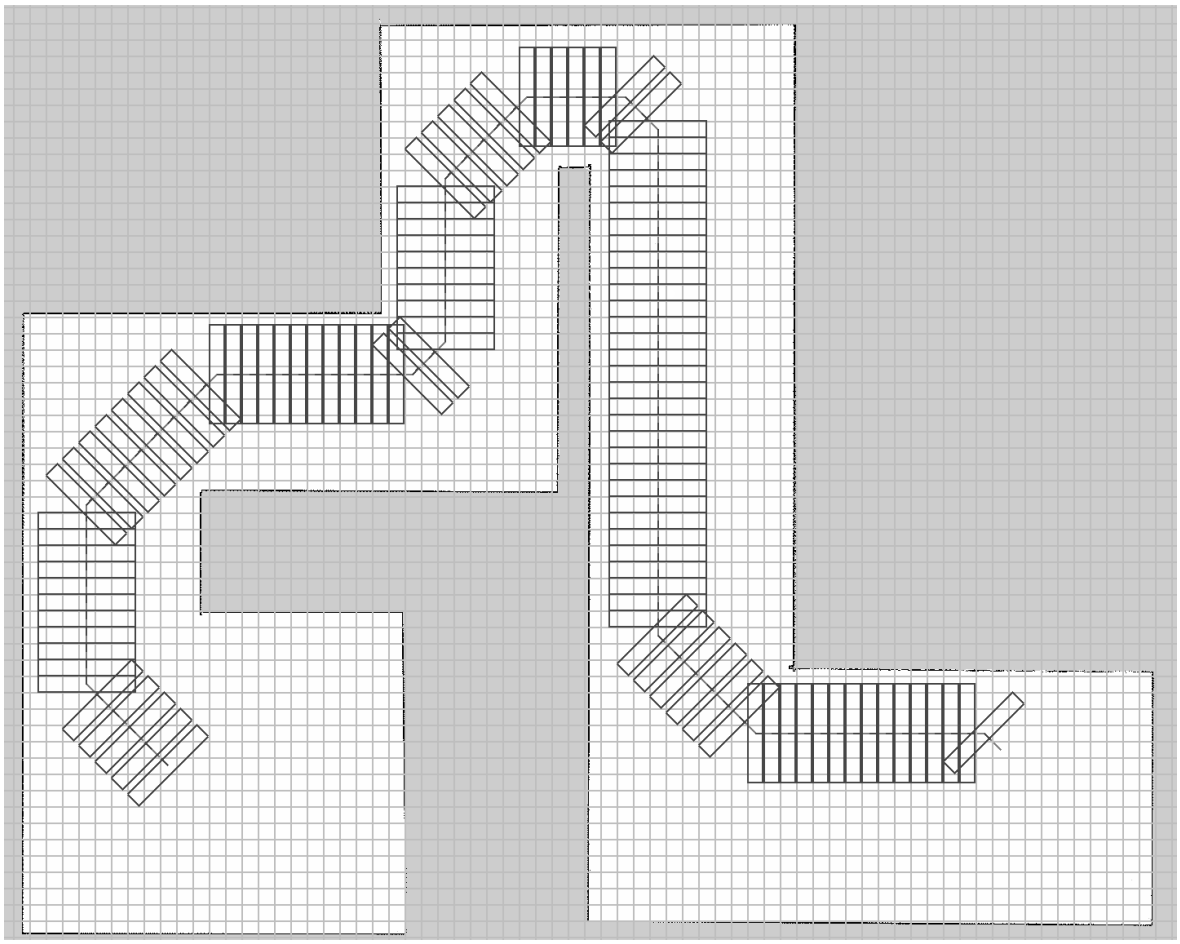


Figure 4.3: Global path example

4.2 Motion planning

Based on the global path given by the previously described path planning algorithm, velocity commands are calculated using dynamic window approach [25], [26]. The dynamic window approach is a velocity space based local reactive avoidance technique where search for commands controlling the robot is carried out directly in the space of velocities. Trajectory of a robot can be described by a sequence of circular and straight line arcs [25]. The search space is reduced by kinematic and dynamic constraints of the robot to the finite span of velocities around robot's current velocity vector (v_c, ω_c) that can be achieved within the next sampling interval Δt. Velocity tuple (v, ω) from the possible reachable velocities is considered safe if the robot is able to stop along the trajectory generated by velocities (v, ω) before hitting any object that may be encountered along the path. The velocity maximizing a certain objective function $\Gamma(v, \omega)$ is chosen from the reduced set of velocities [25]. The objective function includes a measure of progress towards a goal location, the forward velocity of the robot, and the distance to the next obstacle on the trajectory. By combining these, the robot trades of its desire to move fast towards the goal and its desire to ship around obstacles (which decrease the free space). The combination of all objectives leads to a very robust and elegant collision avoidance strategy.

Since current motion planning algorithm is designed for single differential drive robot, it outputs velocity commands for it. Our structure consists of two robots and cargo on top of them, where each robot is abstracted as the wheel of the structure, so velocity commands that algorithm outputs need to be transformed to velocities that individual robot needs to achieve so the whole structure would do desired motion. Velocities are transformed by following the differential drive kinematic equations [27]. Linear and angular velocities, V and ω respectively, using differential drive kinematics are calculated as:

$$V = \frac{V_L + V_R}{2} \tag{4.1}$$

$$\omega = \frac{V_R - V_L}{b} \tag{4.2}$$

where: V_L and V_R are linear velocities of left and right wheel respectively, in our

configuration - linear velocities of left and right robot, b is the distance between two wheels (distance between two robots in our case), V and ω are desired linear and angular velocity of the whole structure. Solving a system of two linear equations (4.1, 4.2) by unknowns V_L and V_R we get:

$$V_L = V - \frac{b}{2} \cdot \omega \tag{4.3}$$

$$V_R = V + \frac{b}{2} \cdot \omega \tag{4.4}$$

.

which are given to each robot respectively.

The only difference for the calculation of the velocities of real robots, compared to the simulation, is that now the robots are not physically linked to the cargo, so they also need to achieve angular velocities which are calculated as:

$$\omega_l = \omega_r = \omega$$

where ω_l and ω_r are angular velocities of the left and right robot respectively. In the simulation, due to physical linking, $\omega_l = \omega_r = 0$.

In code, local planner is implemented using ROS DWA planner C++ API [28]. The task of the local planner node is to wait for the user to set the goal pose in RViz. When it receives the goal pose, it sends service request to the global planner which calculates global path from the current structure's pose to the wanted goal pose. After receiving the global path, it calculates the velocities (V, ω) that the structure must achieve and publishes them to the corresponding topic until the structure achieves goal pose.

4.3 System architecture and workflow

After all the nodes are successfully launched, admissible cell orientations are calculated and graph is created, system is ready to be used. Using *2D Nav Goal* option inside the RViz, goal pose can be set. After setting goal pose, local planner receives it and sends service request to the global planner node where global planner calculates global path from the current structure's position to the goal pose. If the path is found, global planner

sends response as global path points through which structure needs to pass to reach the goal. If it isn't, then appropriate message is printed in the terminal. After local planner receives global path, it starts calculating velocities which structure needs to achieve and publishes them to appropriate topic. Since the structure itself doesn't have a drive, but its drive consists of two robots that have it, velocities are then transformed for each robot, as stated in the section 4.2.

The abstracted system is depicted in the following figure (Fig. 4.4).

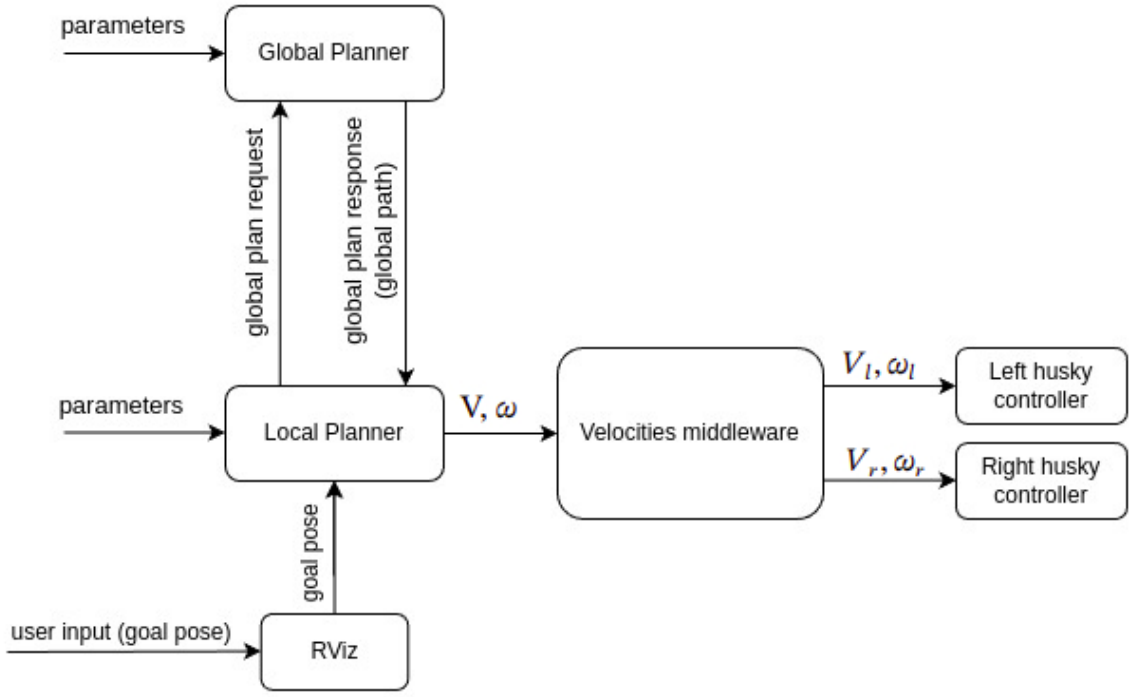


Figure 4.4: Abstracted system

4.4 Additional considerations

In this chapter, initial approaches taken and the problems encountered during these attempts in writing this thesis will be described.

Pioneer 3-DX

The implementation of the thesis began by first creating a Gazebo environment and placing a single Pioneer 3-DX robot in it. For robot's path planning and motion control, ROS *move_base* package [29] was used. move_base is a package that by giving it: robot's sensor, localization and velocity topics, environment information through occupancy map,

and desired goal pose, provides global and local path planning and velocity commands with which robot will reach the desired goal pose collision free.

Global planner was working great because the global path was calculated correctly, localization and low level velocity regulators were also working correctly, but local planner was not, robot was behaving strangely, driving in circles without following global path and reaching the goal. A lot of research was done, we tried many different parameters and different local planners but unfortunately, we could not fix the problem. As stated before, we then chose Husky robot to work with.

move_base's local planners

After switching from Pioneer 3-DX to Husky and making sure that individual Husky can reach any goal, a structure consisting of two Husky robots and cargo was created where cargo is represented as a long wall. Then we tried giving each Husky goal where they would only have to go in a straight path without avoiding obstacles, but for unknown reason local planner would say that it can't find a path, while the path certainly exists. Planners that were tried are: Dynamic Window Approach [28], [25], Trajectory planner [30], Timed Elastic Band [31], [32], but unfortunately problem persisted with each of them. We decided to abandon move_base and its planners and opted for implementation that will be described in the following chapter.

5 Experiments and results

In this chapter, the results of the methods in simulation and also in the real world will be presented.

5.1 Simulation experiments

Experiments in simulation are visualized using Gazebo and RViz. Cargo is represented as a long plank that sits on top of two huskies, huskies and cargo top view simulation setup is shown in the following figure (Fig. 5.1). In simulation, huskies are equipped with laser scanner which is needed for creating a map, localization and navigation.

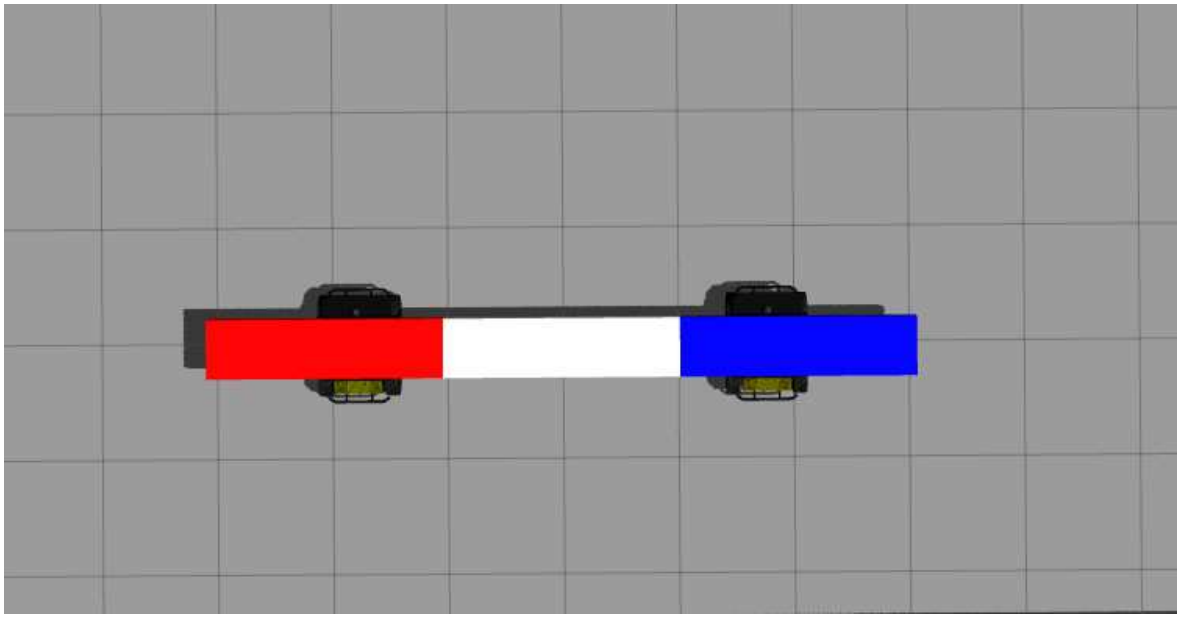


Figure 5.1: Top view simulation setup

.

Cargo and huskies are spawned in custom environtment that can be seen in the figure 5.2.

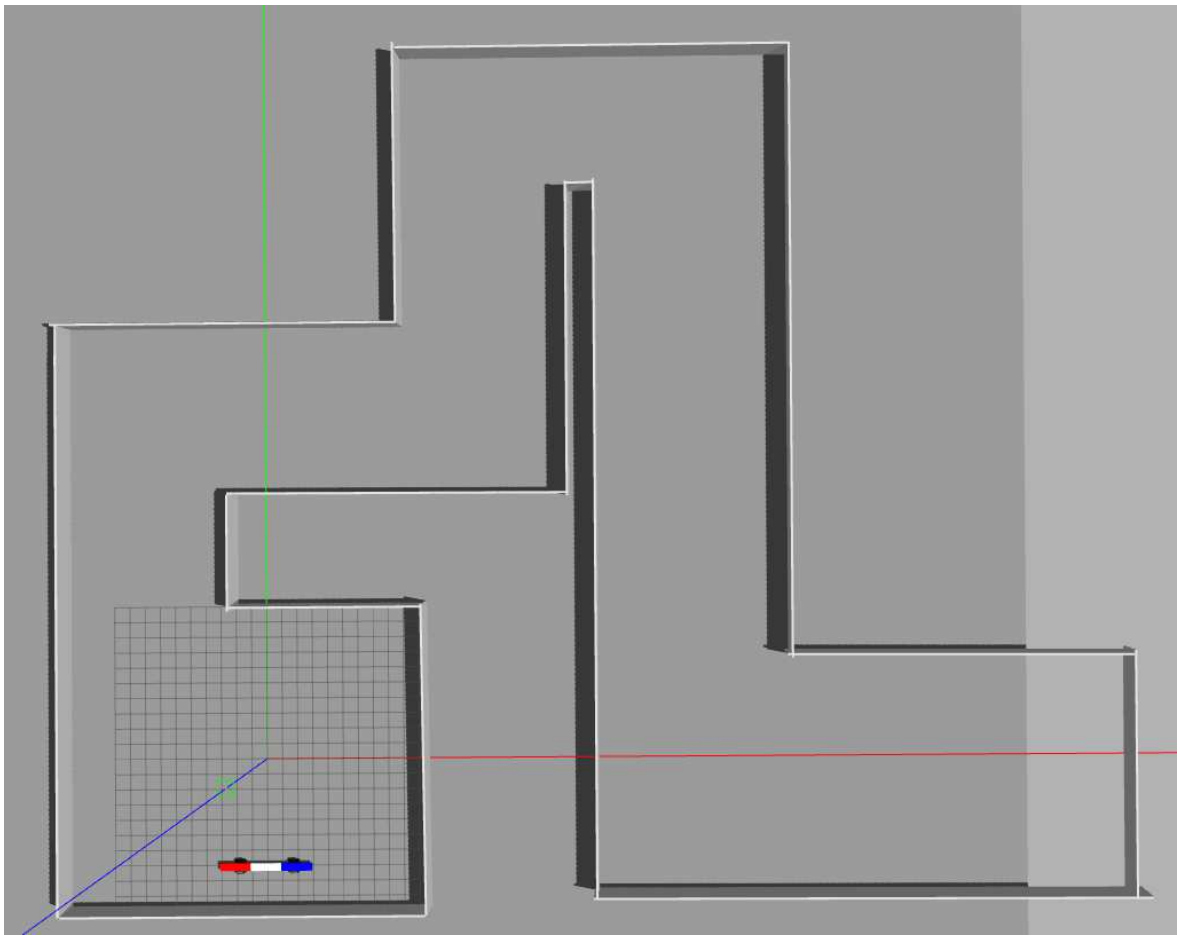


Figure 5.2: Simulation environment

In the figure 5.3, simulation experiment is shown using data published in RViz. Green line represents a global path calculated by the global planner, red line represents also the global path just published by the local planner, blue curve represents the trajectory that structure is moving through with current calculated linear and angular velocities, green rectangle represents structure's footprint.

The following figure (Fig. 5.4) shows the path that the structure traversed through from the beginning to the end of the map, proving that the algorithms work successfully.

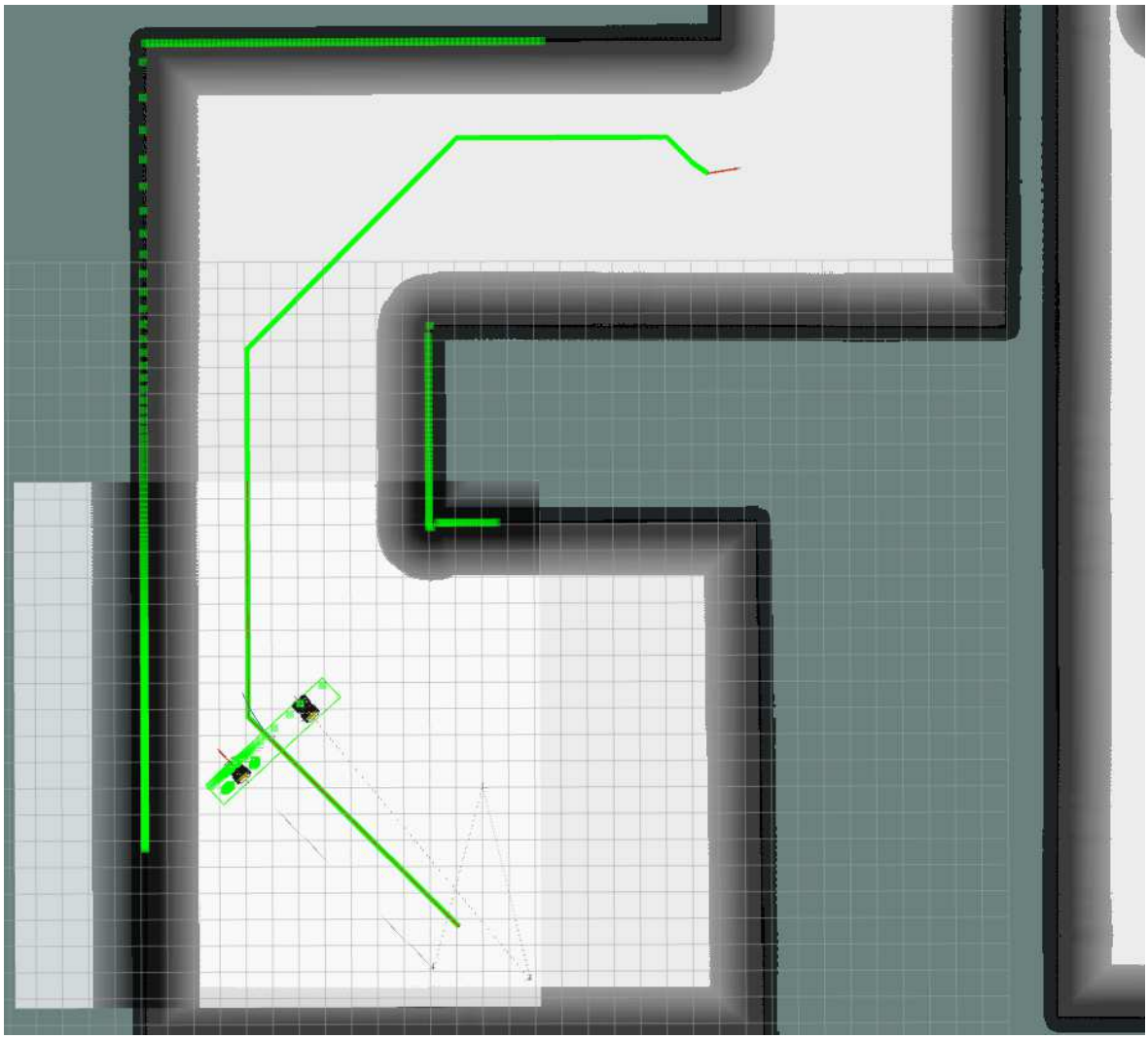


Figure 5.3: RViz simulation

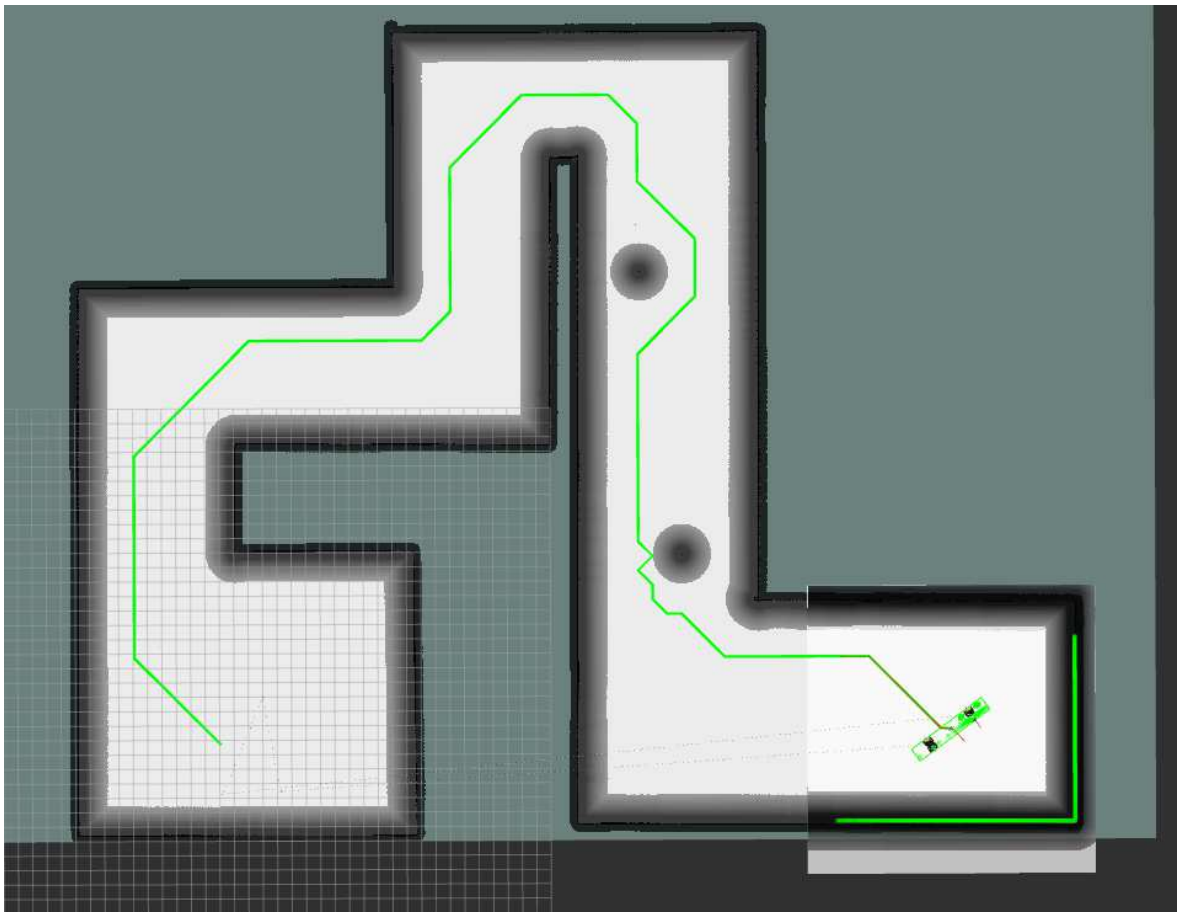


Figure 5.4: Structure successfully traversed from start of the map to the end

5.2 Real world experiments

Real world experiments' setup is shown in the figure 5.5. Since laptops need to be on top of the huskies to control them, there isn't much space to put cargo without getting in the way of cameras' field of view, that is why long metal bar is chosen to represent cargo. It doesn't take much space on huskies, but it enlarges structure's footprint as the real cargo was on top of them.

Figure 5.5: Lab experiment setup

Huskies are equipped with depth cameras which are needed for creating a map, localization and navigation.

Lab environment where experiments were conducted is shown in figure 5.6.

Following figure (Fig. 5.7) depicts huskies' starting pose and environment. Green dots represent left husky's laser measurements while white dots represent right husky's laser measurements.

After setting a goal pose, algorithm calculates global path and local planner successfully drives structure to the goal pose without colliding with obstacles. Example of information published in RViz after setting goal pose in real environment is shown in figure 5.8, while figure 5.9 depicts structure's pose after reaching given goal pose.

Figure 5.6: Lab environment

Figure 5.7: RViz environment information

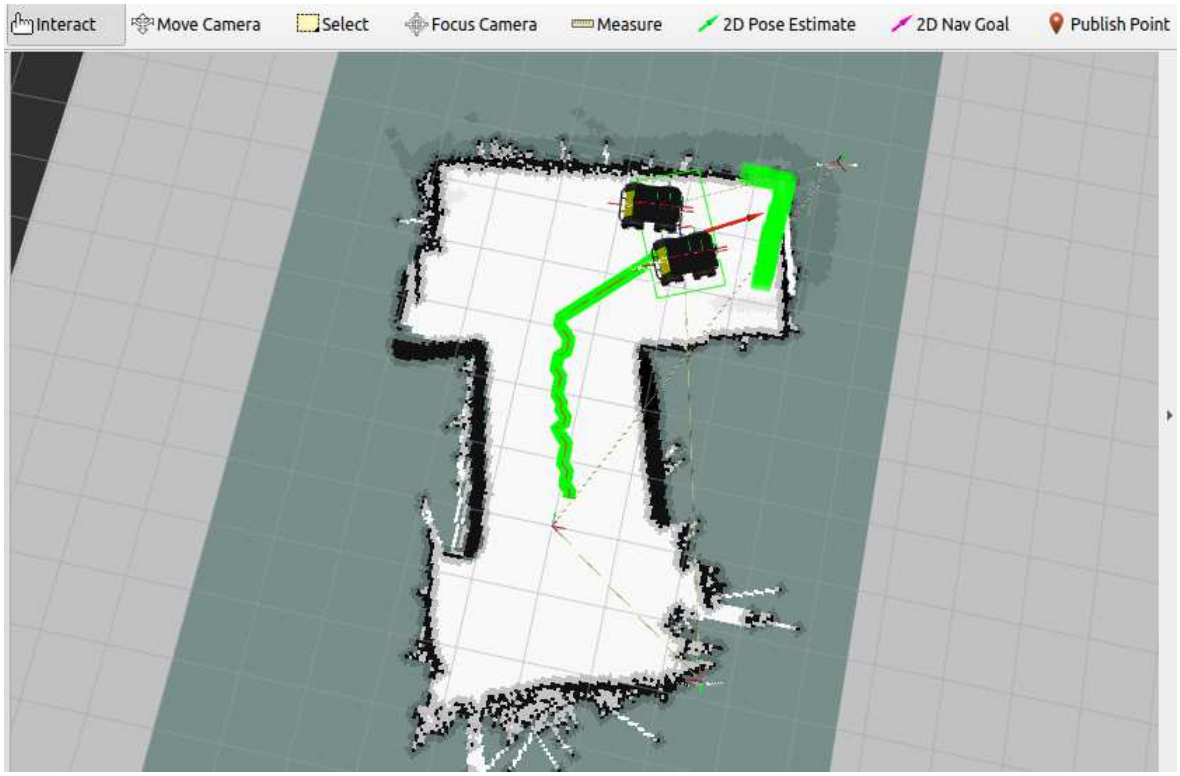


Figure 5.8: RViz information after structure arrived at the goal pose in real world experiment

Figure 5.9: Structure's pose after reaching its given goal pose

6 Discussion

As shown in the previous chapter (Chap. 5), the algorithms work successfully and if there is a path from the current pose of the structure to the target pose, the global path planner will successfully find the path, and the local planner will calculate the velocities with which the structure will successfully drive to the target pose.

In the simulation, everything works perfectly. For ease of testing, the ground truth pose of the robot and structure provided by Gazebo is used. Laser scanner is used for localization and navigation which is more precise than the camera. In real-world testing, the AMCL package [33] is used for localization. In the real world, some problems occur that are not present in the simulation. Since a laser scanner is not used to create the map, but a depth camera whose pointcloud measurements are converted into laser scans - there are deviations and imperfections of the measurements, and because of this the map is not created perfectly as in simulation. Also, for localization and navigation, camera measurements converted into laser scans are used which are less correct compared to the real laser that was used in the simulation. Huskies can skid, so there are imperfections in their odometry. Also, it is difficult to perfectly accurately determine the initial pose of the structure in testing. For easier retesting, black rectangles are marked into which robots are put for their staring pose, which can be seen in figure 5.5, but robots are never put in the exact same pose from test to test because it is impossible without additional sensors to position them perfectly so that they would be always in the same starting pose. Due to the mentioned problems, there is a discrepancy in the pose of the structure in the real world and in the pose in which the localization algorithms think the structure is located, and because of this - sometimes the structure thinks it is further forward than it really is and then starts to turn prematurely, which results to structure passing too close to the obstacles.

The mentioned problems could be solved by better localization, for example using OptiTrack or some other sensor. However, despite the described problems, the path and motion planning algorithms work well also in a real environment where the structure successfully arrives without collisions from its start to the target pose, and with the solution to the localization problem, the behavior and motion of the structure would be even better and there would be almost no differences between the simulation and the real world behavior.

7 Conclusion

This thesis proposes solution to a problem where two mobile robots need to carry a large load, that is too large for a single robot, from its starting pose to the goal pose while bypassing obstacles and maneuvering through narrow passages. Algorithm for global path planning of an arbitrary shape *robot* (structure) with differential drive, as well as a motion control algorithm for controlling the structure, are presented. The methods are implemented within Robot Operating System (ROS) and validated in a simulation and real environment where the algorithms have been shown to work successfully. The structure consists of two robots carrying a large load and together they form a kinematic chain for which the optimal global path to the desired goal is calculated. After the global path is calculated, the algorithm sends control signals to each robot separately and the structure achieves a motion as if it had only one drive and not that it consists of two robots each having its own drive.

In regards to prior researchers' work which mainly use a leader-follower configuration, main contribution of this method is that velocities of one robot are calculated independently of the other robot, so robots move independently of one another, they don't even know about the existence of another robot. This method makes setup of robots and cargo much easier, where everything that would be needed is to set robots' starting poses and define the footprint that structure occupies, after that all that is required is to set the goal pose to which the structure will successfully arrive, if the path exists.

The proposed approach can be widely applied in areas such as logistics and industrial robotics, contributing to the optimization of the load transfer process and improving the efficiency of autonomous robots in work environments.

References

[1] “Robotics,” 2024, last accessed 20 June 2024. [Online]. Available: https://en.wikipedia.org/wiki/Robotics

[2] J. Hu, P. Bhowmick, and A. Lanzon, “Group coordinated control of networked mobile robots with applications to object transportation,” *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 8269–8274, 2021. https://doi.org/10.1109/TVT.2021.3093157

[3] J. Hu, P. Bhowmick, I. Jang, F. Arvin, and A. Lanzon, “A decentralized cluster formation containment framework for multirobot systems,” *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1936–1955, 2021. https://doi.org/10.1109/TRO.2021.3071615

[4] J. Alonso-Mora, S. Baker, and D. Rus, “Multi-robot formation control and object transport in dynamic environments via constrained optimization,” *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017. https://doi.org/10.1177/0278364917719333

[5] D. Mohan and A. Vivek, “Navigation of two wheeled mobile robots cooperatively carrying an object,” in *2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT)*, 2017, pp. 1–7. https://doi.org/10.1109/ICCPCT.2017.8074218

[6] A. Yamashita, T. Arai, J. Ota, and H. Asama, “Motion planning of multiple mobile robots for cooperative manipulation and transportation,” *IEEE Transactions on Robotics and Automation*, vol. 19, no. 2, pp. 223–237, 2003. https://doi.org/10.1109/TRA.2003.809592

[7] T. Machado, T. Malheiro, S. Monteiro, W. Erlhagen, and E. Bicho, “Multi-constrained joint transportation tasks by teams of autonomous mobile robots using a dynamical systems approach,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3111–3117. https://doi.org/10.1109/ICRA.2016.7487477

[8] ROS.org. Open Robotics, “ROS - Robot Operating System,” 2024, last accessed 20 June 2024. [Online]. Available: https://www.ros.org/

[9] ——, “Understanding ROS Nodes,” 2024, last accessed 20 June 2024. [Online]. Available: http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes

[10] ——, “Messages,” 2024, last accessed 20 June 2024. [Online]. Available: http://wiki.ros.org/Messages

[11] ——, “Understanding ROS Topics,” 2024, last accessed 20 June 2024. [Online]. Available: http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics

[12] ——, “Services,” 2024, last accessed 20 June 2024. [Online]. Available: http://wiki.ros.org/Services

[13] ——, “actionlib,” 2024, last accessed 20 June 2024. [Online]. Available: http://wiki.ros.org/actionlib

[14] gazebosim.org. Open Robotics, “About Gazebo,” 2024, last accessed 20 June 2024. [Online]. Available: https://gazebosim.org/about

[15] ROS.org. Open Robotics, “Tools/3D Visualization: RVIZ,” 2024, last accessed 20 June 2024. [Online]. Available: http://wiki.ros.org/Tools

[16] Clearpath Robotics, “HUSKY - UNMANNED GROUND VEHICLE,” 2024, last accessed 20 June 2024. [Online]. Available: https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/

[17] Cyberbotics, “Adept’s Pioneer 3-DX,” 2024, last accessed 20 June 2024. [Online]. Available: https://www.cyberbotics.com/doc/guide/pioneer-3dx?version=R2021a

[18] Intel® RealSense™, “Intel® RealSense™ Depth Camera D435i,” 2024, last accessed 20 June 2024. [Online]. Available: https://www.intelrealsense.com/depth-camera-d435i/

[19] Docker, “Docker overview,” 2024, last accessed 20 June 2024. [Online]. Available: https://docs.docker.com/guides/docker-overview/

[20] NetworkX, “NetworkX - Network Analysis in Python,” 2024, last accessed 20 June 2024. [Online]. Available: https://networkx.org/

[21] E. G. Tsardoulias, A. Iliakopoulou, A. Kargakos, and L. Petrou, “A review of global path planning methods for occupancy grid maps regardless of obstacle density,” *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 829–858, Dec 2016. https://doi.org/10.1007/s10846-016-0362-z

[22] ROS.org. Open Robotics, “Global Planner,” 2024, last accessed 20 June 2024. [Online]. Available: https://wiki.ros.org/navigation/GlobalPlanner

[23] M. Đakulović, C. Sprunk, L. Spinello, I. Petrovic, and W. Burgard, “Efficient navigation for anyshape holonomic mobile robots in dynamic environments,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2644–2649. https://doi.org/10.1109/IROS.2013.6696729

[24] E. W. Dijkstra, “A note on two problems in connexion with graphs,” in *1959 Numerische Mathematik 1*, 1959, pp. 269–271.

[25] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robotics Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997. https://doi.org/10.1109/100.580977

[26] M. Seder and I. Petrovic, “Dynamic window based approach to mobile robot motion control in the presence of moving obstacles,” in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 1986–1991. https://doi.org/10.1109/ROBOT.2007.363613

[27] G. Klancar, A. Zdesar, S. Blazic, and I. Skrjanc, *Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems*, 1st ed. USA: Butterworth-Heinemann,

2017, ch. 2.2.1.

[28] ROS.org. Open Robotics, “DWAPlannerROS,” 2024, last accessed 20 June 2024. [Online]. Available: https://wiki.ros.org/dwa_local_planner

[29] ——, “move_base,” 2024, last accessed 20 June 2024. [Online]. Available: http://wiki.ros.org/move_base

[30] ——, “TrajectoryPlannerROS,” 2024, last accessed 20 June 2024. [Online]. Available: https://wiki.ros.org/base_local_planner

[31] ——, “Timed-Elastic-Band local planner,” 2024, last accessed 20 June 2024. [Online]. Available: https://wiki.ros.org/teb_local_planner

[32] C. Rösmann, F. Hoffmann, and T. Bertram, “Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control,” in *2015 European Control Conference (ECC)*, 2015, pp. 3352–3357. https://doi.org/10.1109/ECC.2015.7331052

[33] ROS.org. Open Robotics, “amcl,” 2024, last accessed 20 June 2024. [Online]. Available: https://wiki.ros.org/amcl

Abstract

Coordinated motion planning and control of mobile robots for joint transportation of large objects

Borna Paro

In this thesis a method is presented for coordinated motion planning and control of two autonomous mobile robots for the joint transport of cargo that is too large for one robot. The method was implemented within the Robot Operating System (ROS) environment and validated in simulation and real world scenarios. The method consists of two robots having large cargo on top of them where they together form a new kinematic chain. Then, the optimal path is planned for the entire differential drive kinematic chain that ensures the transport of the cargo to the desired target location collision free. After that, the implemented algorithm sends control commands to each robot separately, which achieves the coordinated execution of the planned path. The proposed approach can be widely applied in areas such as logistics and industrial robotics, contributing to the optimization of the load transfer process and improving the efficiency of autonomous robots in work environments.

Keywords: autonomous mobile robots; ROS; Python; C++; path planning; coordinated motion planning; differential drive; kinematic chain

Sažetak

Koordinirano planiranje gibanja i upravljanje mobilnim robotima za zajednički prijevoz velikog tereta

Borna Paro

U ovom radu prikazana je metoda za koordinirano planiranje gibanja i upravljanje dvoje autonomnih mobilnih robota za zajednički transport tereta koji je prevelik za jednog robota. Metoda je implementirana unutar okruženja Robot Operating System (ROS) i testirana u scenarijima simulacije i stvarnog svijeta. Metoda se sastoji od dva robota koji imaju veliki teret na sebi gdje zajedno tvore novi kinematički lanac. Zatim, optimalni put je izračunat za cijeli kinematički lanac diferencijalnog pogona koji osigurava prijevoz tereta do željenog cilja bez sudara s preprekama. Nakon toga, implementirani algoritam šalje upravljačke naredbe svakom robotu zasebno, čime se postiže koordinirano izvođenje planiranog puta. Predloženi pristup može biti primijenjen u područjima kao što su logistika i industrijska robotika, pridonoseći optimizaciji procesa prijenosa tereta i poboljšanje učinkovitosti autonomnih robota u radnim okruženjima.

Ključne riječi: autonomni mobilni roboti; ROS; Python; C++; planiranje puta; koordinirano planiranje gibanja; diferencijalni pogon; kinematički lanac