UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Technology
Robotics and Computer Engineering Curriculum

Md. Maniruzzaman

Object search and retrieval in indoor environment using a Mobile Manipulator

Master's Thesis (30 ECTS)

Supervised By: Associate Professor Arun Kumar Singh

Tartu 2022

Resümee/Abstract

Objektide otsimine ja teisaldamine siseruumides mobiilse manipulaatori abil

Roboteid nähakse üha enam teenindusagentidena kontorites ja kodudes. Paljudes riikides, kus keskmine elanikkond vananeb, saab roboteid kasutada ka eakate hooldamiseks. Käesolevas väitekirjas uuritakse ühte sellist võimalust, kasutades mobiilset manipulaatorrobotit. Sellistel robotitel on liikuv baas, mis võimaldab liikuda ühest kohast teise ning samuti ka käsi, millega saab esemeid valida ja paigutada. Käesolevas väitekirjas käsitletakse probleemi, kus mobiilne manipulaator peab otsima keskkonnas objekti ja viima selle teise kohta. Optimaalne objekti otsimine on sõnastatud populaarse rändkaupmehe probleemi abil (TSP), mis arvutab optimaalse järjestuse, milles robot saab külastada kõiki võimalikke asukohti, kus objekt võib olla. Eelnev teave tõenäolisemate asukohtade kohta tuuakse sisse skaleerides TSP graafiku servade kaalu asukoha tõenäosuste kaudu. Lõputöö kombineerib TSP väljundit navigatsiooni ja manipulatsiooni planeerimisega, mis on ehitatud Robot Operating Systems (ROS) peale, et luua täielik objektide otsimise ja leidmise protsessiahel. Lõputöö tulemused on valideeritud nii simulatsiooni kui ka tegelike riistvarakatsetega.

CERCS:T120 Arvutitehnoloogia; T125 Automatiseerimine, robootika

Märksõnad:TSP, tõenäosus, ROS, mobiilne manipulaator, objektiotsing, haaramine.

Object search and retrieval in indoor environment using a Mobile Manipulator

Robots are increasingly viewed as service agents in offices and homes. In many countries where the average population is aging, robots can be used for elderly care. This Thesis explores one such possibility using a mobile manipulator robot. Such robots have a mobile base to move from one place to another and an arm to pick and place objects. This Thesis considers a problem where the mobile manipulator needs to search for an object in an environment and bring it to some location. The optimal object search is formulated in terms of the popular traveling salesman problem (TSP) that computes the optimal sequence in which the Robot can visit all the possible locations where the object can possibly be. Prior information about the more likely locations is brought in by scaling the edge-weight of the TSP graph through the probabilities of the location. The Thesis can combine the output of TSP with navigation and manipulation planning built on top of Robot Operating Systems (ROS) to build the complete object search and retrieval pipeline. The results of the Thesis are validated both in simulation and actual hardware experiments.

CERCS:T120 Computer technolog;T125 Automation, robotics,

Keywords: TSP, Probability, ROS, Mobile Manipulator, Object Search, Grasping.

Contents

List of Figures

1 Introduction

Robotic systems had an immense socio-economic impact. Conventionally, due to their incredible endurance, speed, as well as accuracy, they have been mostly deployed in industrial environments for operations on manufacturing floors [1]. But due to the rapid improvements in sensors, compute power, and artificial intelligence algorithms, robots are gradually making the transformation from factory floors to applications like product delivery, diagnosis, disinfection, logistics, manufacturing, medical assistance, etc [2], [3] [4]. The rise of e-commerce has also prompted the creation of warehouse robots, which are utilized for automated storage and product retrieval [5]. Robots are also being increasingly deployed in hospitals, airports, hotels, restaurants, etc as service agents to assist humans. Specifically, autonomous mobile manipulation is expected to be a prominent subject in the next generation of robotics applications, and it is envisioned that these robots will play an essential role as helpers in homes, care facilities, and other settings where people interact with robots. [6]

Motivation: The proposed thesis is motivated by the application of mobile manipulators as assistive agents in indoor spaces such as offices, homes, malls, airports, hospitals etc. A typical scenario as envisioned in this thesis is presented in Fig.1.1. A human asks the robot to search for a coffee mug and points out that it could be located in one of the then possible tables in the office. The robot is then required to locate the object, grasp it and then bring it back to the human. The setting of Fig.1.1 can also be adapted to an application where the robot provides assistance at homes to elderly people with limited mobility. We formally define our problem statement next.

1.1 Problem Statement

The problem statement of this thesis can be formally defined in the following manner:

Given a mobile manipulator, a map of the environment, a mobile manipulator is required to search and retrieve an object from the environment. Additionally, the list of possible locations of the object in the environment is provided as an additional input.

The problem statement can be decomposed into following sub-problems

Autonomous Navigation: To perform object search, the mobile manipulator would need the ability to autonomously navigate in an environment using prior built maps in the form of occupancy grid maps. Moreover, it should be able to use LIDAR scans to avoid previously unseen obstacles. Another essential requirement is to compute the optimal way to visit all the possible locations where the object can possibly be. In this thesis, this will be referred to as the *Route Planning.* Clearly, there could be an infinite number of ways to visit these locations. For efficient object search, the robot should be able to choose one that takes the minimum amount of time. Alternately, the robot can look to minimize the total distance traveled, which will indirectly minimize the total time.

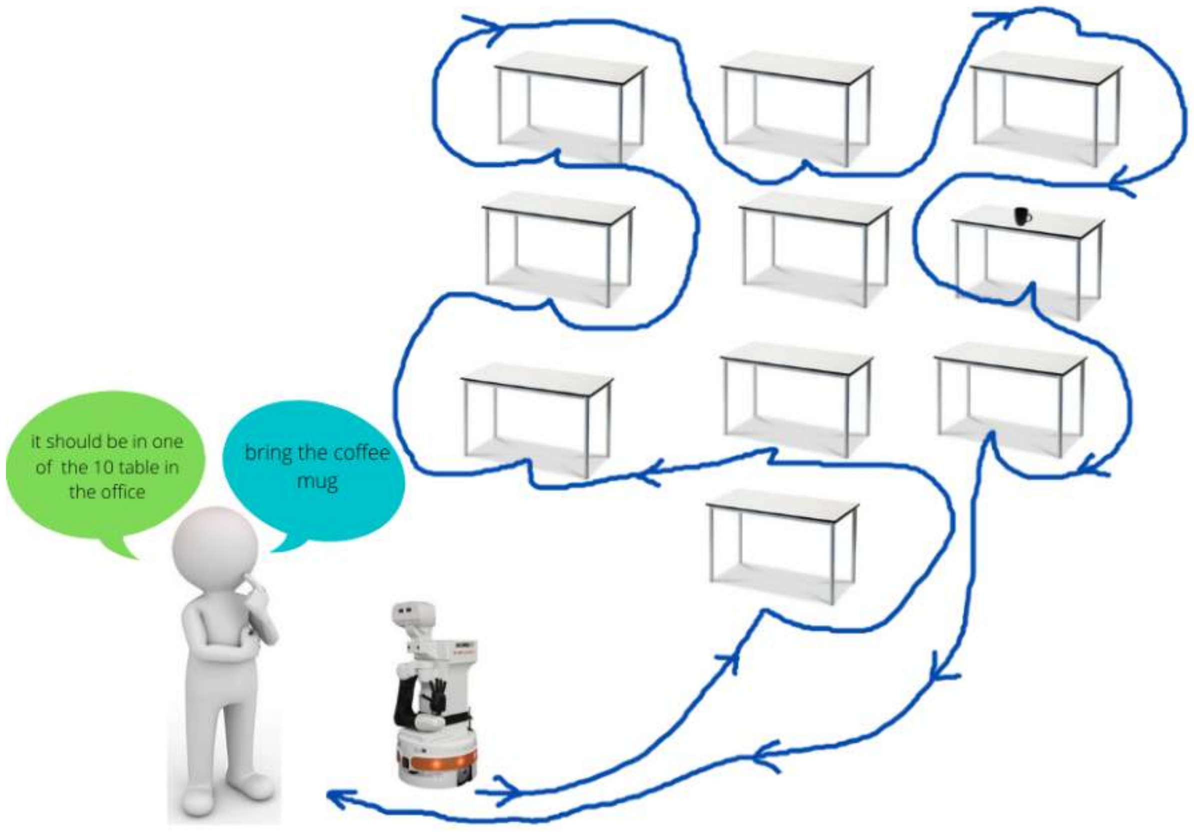


Figure 1.1. Problem statement graphe

Manipulation and Grasping: Once the robot has located the object, it needs to grasp it and bring it to the specified location. Thus, the second component of our problem statement corresponds to grasping and manipulation. The robot should be able to precisely compute the pose of the object relative to itself and then use this information to plan its arm motions for grasping. Furthermore, during grasping, the robot arm should avoid collisions with the environment and the grasped object. For example, in this thesis, the object is assumed to be placed on a table or shelf. So, the robot, while performing the grasping operation, should avoid collisions with the table.

1.2 Contribution of the Thesis

The thesis presents a reliable and repeatable solution pipeline for object search and retrieval in indoor environments and validates it both in simulations and real-world hardware experiments. The overall pipeline can be divided into the following contributions.

- The thesis formulates the object search as a graph-search problem where the object locations are treated as vertices of the graph, and the distance between the locations are modeled through edge-weights. Consequently, the optimal route planning is formulated as the Travelling Salesman Problem (TSP) problem. The thesis also shows how prior information about more likely positions can be brought into consideration by scaling the edge-weights of the TSP.

- The thesis couples the optimal sequence output of the TSP with local trajectory planning for autonomous collision-free navigation and object search.
- The thesis integrates vision-based pose estimation and manipulation planning for object grasping and retrieval.

1.3 Thesis Layout:

Here is a simple layout of the Thesis.

- In the literature review chapter2, we have described some previous works on assistive robotics, object search, TSP, and its use in robotics, as well as works done for picking and placing an object using a mobile manipulator.
- In Requirements 3, we introduce TIAGo, which was our chosen Robot, as well as disclose some of the basic knowledge, and research tools used to do this work.
- A complete research overview is explained with a flowchart at the research overview inside the methodology chapter4. Route Planner, Navigation stack, and Object search and Retrieval Pipeline are some of the methods we follow to complete our objectives are described in detail in this chapter.
- We showed our testing environment and our obtained result for simulation and hardware implementation in the Result chapter5.
- We have wrap-up our whole report in conclusion 6 and also described some future work we can do.

2 Literature Review

The literature review is divided into the following parts. Since the core application area of the thesis is in assistive robotics, a survey of the existing assistive robots is presented. Second, a review of autonomous object searches through mobile robots followed by a review of the existing approaches for solving the TSP problem.

2.1 Existing works on using Assistive Robots

Mobility devices customized with unique hardware, software, and peripherals are examples of embodied assistive technology that help persons with impairments access information technologies. Although these systems are useful, they generally only have one purpose and are not very intelligent. Some of them are more advanced than they can understand the user's demands using sensors and cameras. Smart homes, virtual assistants, and ambient assisted living (AAL) settings are all examples of this. Socially assistive robotics (SAR) are developed to bring people close to robot technology by increasing the scope of human interaction [7]. PHAROS is a social robot that monitors and assesses daily physical activity in the user's house. To appropriately recognize and assess the workout performance, machine learning methods, especially a convolutional neural network (CNN) and a recurrent neural network, are applied. It also has a recommendation algorithm that produces a workout for the user every day, ensuring that they are performing the right amount of physical activity [8], [9]. HOBBIT is a robot designed to make elderly people feel comfortable in their own homes. With this goal in mind, the robot can autonomously navigate around the user's apartment, going wherever they want, picking up objects from the ground, bringing a specific object, learning new objects to be found in the future, calling in an emergency, providing entertainment games, and reminding the user to take their medication [10]. The EU project RAMCIP made a robot that can help older people and people with dementia or mild cognitive impairments (MCI). While this robot may be used to monitor a person's daily routine, it also contains functions that stimulate physical and mental activity, such as making sure the stove is turned off after preparing a meal or making sure the lights are on while strolling at night, picking up fallen or abandoned items and storing them in a secure location, reminding users about their meditation, and bringing the corresponding materials [11]. Rudy's [12] major goal is to help individuals in nursing homes and other healthcare institutions. Remote patient monitoring (RPM), medication reminders, and prescription distribution are just some of the telemedicine features available on this robot. It also has a social component that, along with its inviting appeal, draws people in. In reality, since loneliness is a big concern among the elderly, the system's social connections are the most valued feature. Stevie II, a socially assistive robot, assists carers in long-term care settings, enabling them to focus on person-centered duties. Its features include medication reminders and quizzes, and activities to keep patients mentally active. Enhanced expressive skills and a well-defined social component are employed to achieve this [13]. There are many more humanoid and non-humanoid robots that help to treat autism spectrum disorder (ASD). Some of them are Zeno R-50, Nao, Pepper, KASPAR, Keepon, Popchilla, PABI, Pleo, Robota, i-Sobot, Tito, GIPY-1, HOAP-3, Labo-1, Ifbot, Cosmobot, Ryan Companionbot [7].

The object search and retrieval problem pipeline developed in this thesis can be applied to any of the assistive robots discussed above since the proposed pipeline is general and builds on the popular ROS pipeline.

2.2 Existing works on using Mobile Robots for Object Search

A dynamic object search algorithm has been developed in [14] and tested in a natural home-like environment for searching objects like laptops. A probabilistic model is employed in the cited work to increase the search efficiency. During disasters, search and rescue operations are usually dangerous and time-consuming for human firefighters. With the help of an RF ID tag and by coordinating the local maps made by the robot swarm, a system of mobile robots that work on their own can make a global map. To locate each mobile robot on the global map, a trilateration-based localization system has been devised.[15] Semantic Linking Maps (SLiM) as a Conditional Random Field has made on a concept that keeps belief over the location of a target object as well as the locations of landmark objects while taking into account the probabilistic spatial relationships between objects. A hybrid search strategy is also introduced to choose the next best view poses for searching for the target object based on the belief that has been kept. [16] Considering the health issue of searching for the radioactive source, Autonomous Searching Mobile Robots have been tested, where a partially observable Markov decision process (POMDP) is used to detect the radioactive materials with sensor data, and the Bayesian framework is used to estimate the location of the radioactive materials.[17]A UAV and a ground vehicle collaborate together for a search and rescue mission, where the faster UAV does a quick search for the object and generates a map of an unknown area. The UAV also find some possible location of the object that the autonomous ground vehicle can visit.[18]

2.3 Travelling Salesman Problem

Brief Historical Review of TSP

Mathematicians and computer scientists alike have paid close attention to the traditional TSP. So we can get an idea from its history of how scientists emphasize TSP. When TSP first came forward, it was considered a mystery, but according to the historian, in 1832, a traveling problem between Germany and Switzerland had been discussed without any mathematical implementation [19]. Mathematician W.R. Hamilton (Iris) and Thomas Kirkman(British) gave a mathematical model of TSP inside graph theory in the 19th century [20] when they were solving the hamilton cycle. The brute force algorithm was proposed by Karl Menger in 1930 to solve the TSP. In the same year, TSP considered a mathematical model when Merrill M. Flood was trying to solve the bus trip problem [21][22]. The cutting plane technique was implemented with a branch and bound algorithm in 1950-1960 by Dantzig, Fulkerson, and Johnson for solving [21]. The very first practical solution provided in 1959 was known as Beardwood–Halton–Hammersley theory [23]. In 1960 branch bound combined with a lower bound and minimum spanning tree was declared to solve the TSP [21]. Hamiltonian Cycle problem was an NP-complete proven by Richard M.karp in 1972, who proposed that the TSP is NP-hard. The Christofides-Serdyukov algorithm was proposed by Christofides, and Serdyukov and they had given the worst-case scenario concept in their algorithm which makes the TSP optimal solution simple and quicker

[24]. The algorithm was unchanged since 1976, and an improvement appeared in 2011 [25]. Later this improvement called approximation algorithm became full(metric) TSP [26], [27]. TSP was successfully applied in 2393 cities between 1970 to 1980 by Padberg, Grötschel, Rinaldi, and others where they applied cutting and branch and bound algorithm. The first programmed TSP is known as the Concorde program, developed by Applegate, Bixby, Chvátal, and Cook in 1990. TSPLIB was published in 1991 and developed by Gerhard Reinelt. 85900 cities' optimal tour was successfully done in 2006 by Cook and others.
TSP in Robotics There are several algorithms have been proposed and designed to solve the TSP; some of them are Simulated Annealing(SA), Tabu Search(TS), Hill Climbing(HC), Genetic Algorithm(GA), Evolutionary Strategy(ES), Evolutionary Programming(EP), Ant Colony Optimization(ACO), Particle Swarm Optimization(PSO), Artificial Bee Colony(ABC), Fruit Fly Optimization Algorithm(FOA) and so on. Every algorithm has some advantages and disadvantages, SA, TS, and HC are simple, flexible, and easy to implement, but they easily fail into local optimum. GA, ES, and EG have the capability of self-organization, self-adaption, self-learning but the demerits of this algorithm are fixed mode, increased parameter space and often poor quality local minima solution. ACO, PSO, ABC, and FOA have high robustness of intelligence, and information interaction, group collaboration but may fall into local optimum. Ant colony algorithm has poor diversity and slow convergence for path planning problems, MRCACO (multi-role adaptive collaborative ant colony optimization) has been developed for solving this problem, and this algorithm proves its feasibility in solving the path planning problem[28] ITSP(Indoor Traveling Salesman Problem) was developed based on branch and bound (B&B) and Dijkstra algorithm with the concept of applying TSP in an indoor environment. This ITSP path planning has been tested on a six-floor shopping mall [29]. In [30], for planning a path Traveling salesman problem with the circular neighborhood(TSPCN) has been designed for the laser welding robot where a UAV supplied the data used to solve the problem.

2.4 Mobile Manipulator Pick and Place

HERB is one of the first robots developed to search and retrieve objects in household environments. This project's goal was to carry out routine tasks in-home or office-like environment [31]. In [32], part pickup and transportation tasks in a warehouse-like environment have been performed by a two-finger gripper mobile manipulator, where the robot base moves in a predetermined path, and the collision-free movement of the arm is ensured. In [33], a hierarchical search-based method for mobile manipulator pick and place. An end-to-end pipeline of visual perception, motion planning, and grasping is implemented on a mobile manipulator in [34] for object search. A "waiter" robot is proposed in [35] for serving drinks. OpenCV is used to estimate the posture of a textured object in real-time. It uses PnP + Ransac to calculate the location. In addition, a Linear Kalman Filter is used to reject undesirable postures. This technique has the drawback of only working with textured items and requiring a 3D representation of the item.

3 Requirements

3.1 Tiago

A brief explanation of the robot will be presented in order to offer a better understanding of the robot's many traits and features. The robot contains numerous distinct components of interest for this project in terms of hardware. From top to bottom, they are described. The first component is a movable head that houses an RGB-D camera. To conduct tilt and pan rotations, the head may be moved utilizing two motors on its base. Without moving the robot, these motions may be utilized to focus the camera view range to various places. This enhances the camera's perception area's viewing range. The robot has a stereo microphone for communication and a speaker which can replicate various sounds just below the head, at the base of the neck. The 7 degrees of freedom arm positioned on the robot's chest is the most significant feature when this comes to the robot's torso. As an inverted 7DOF industrial arm, the first joint may rotate the shoulder over the ground plane. This robot's mechanical restrictions have been altered to display a left-handed configuration rather than the conventional right-handed one. A force/torque sensor is built into the end-effector base to monitor the payload on the arm's end. The last tool on the arm is a two-claw gripper bit different than the one shown in Figure 3.1. A prismatic joint is situated on top of the movable base, running from top to bottom. By elevating or lowering the torso, this link may be stretched and retracted to modify the robot's height. Finally, the mobile base is attached to this connection. This is a differential drive mobile base that contains all of the navigational components. To effectively explore the base, it has a belt of LEDs that can be controlled to perform various lights and patterns, a laser range-finder sensor, and two back sonar sensors.

In terms of software, TIAGo runs on a 64-bit version of Ubuntu and employs the RT Preempt real-time framework. Between the software layer and the Ubuntu OS, the robot employs ROS as a robotics middleware. A robotics middleware is a communication layer between the user's software solutions and the various drivers of the operating system used to operate the robot. ROS is a layer of an open-source operating system that provides hardware abstraction, implementation of commonly used functionality, low-level device control, message-passing between processes, and package management. The robot offers a variety of tools and capabilities that are highly beneficial to developers over this ROS layer. There are a variety of packages already loaded in the robot that may be used for a variety of tasks, including commanding the base and detecting human faces using camera photos. The more relevant for this project will be discussed and referenced throughout the report. [35]

3.1.1 Tiago & Safety

Power On

Tiago's arm usually lays on top of the movable base, and in some circumstances, it collides with it in order to avoid it.

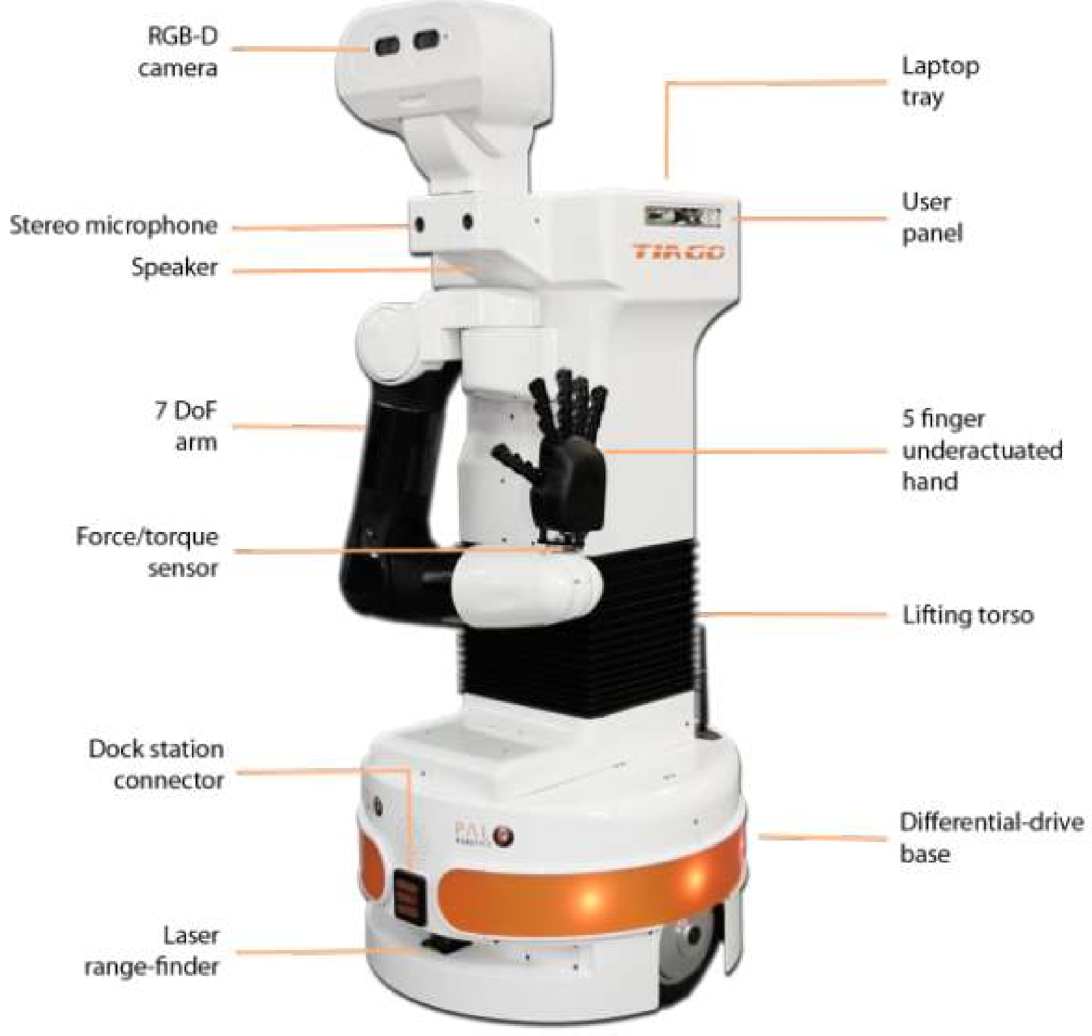


Figure 3.1. Used Components In TIAGo

- either use the Get out of Collision button on the Demos page of WebCommander or export cmd rosservice call /get out of collision ROS MASTER URI=http://tiago-133c:11311
- Using the joystick, raise the torso to its full height.
- Use the Movements tab of the WebCommander to execute the Offer Gripper or (or Offer Hand if you have the Hey5 Hand mounted) action.
- Execute the Home action to fold back the arm and torso into a safe position where no contact with the robot's lap occurs.

Power Off

Special precautions must be taken to prevent banging the arm against the robot's base or the floor, such as:

- Use the WebCommander, for example, to carry out the Home predetermined move.

- Hold the wrist of the robot and wait until the green light indication stops flashing after pressing the On/Off button for 1 second.
- Continue to hold the wrist until the robot turns off. The wrist may then be guided to its resting position on top of the movable base.

Emergency stop

When we apply this emergency stop, the motors will be turned off and the arm will fall down, but the computer will stay on. We must restart the robot by pushing the On/Off button until it stops blinking after releasing the emergency stop button. In a few seconds after this process, the robot's state should be returned to that of before pushing the emergency button.

Fall prevention

It is important to take preventative actions in order to avoid the robot from tumbling over or robot from toppling over.

- When the Hand is in use, no external force should be applied. Expand the figure 3.2(a)
- Do not travel with the arms outstretched, particularly if the body is extended as well.3.2(b))
- Tiago was created with flat floor navigation in mind. (Figure3.2(c)): Do not maneuver on flooring with more than 5% unevenness.
- Avoid traveling near descending steps(ex: stairs) since the TIAGo's laser range-finder will miss this and the robot will tumble down.
- It is strongly advised that you move with your arm folded and your body at a low extension, like in the standard Home configuration (see figure 3.2(d)). This position has the following benefits:
  - Reduces the robot's footprint, which reduces the chances of the arm colliding with the surroundings.
  - Because the robot's center of mass is near to the robot's core and maintained low, it ensures the robot's stability.

Avoid collisions

- The arm's greatest reach without the End-Effector is 86 cm, therefore there must always be enough space in this diameter.
- The startup extras' collision detector node, purports to provide an example of how to achieve safety via feedforward current control. A disadvantage is that it does not function when Whole Body Control is on.

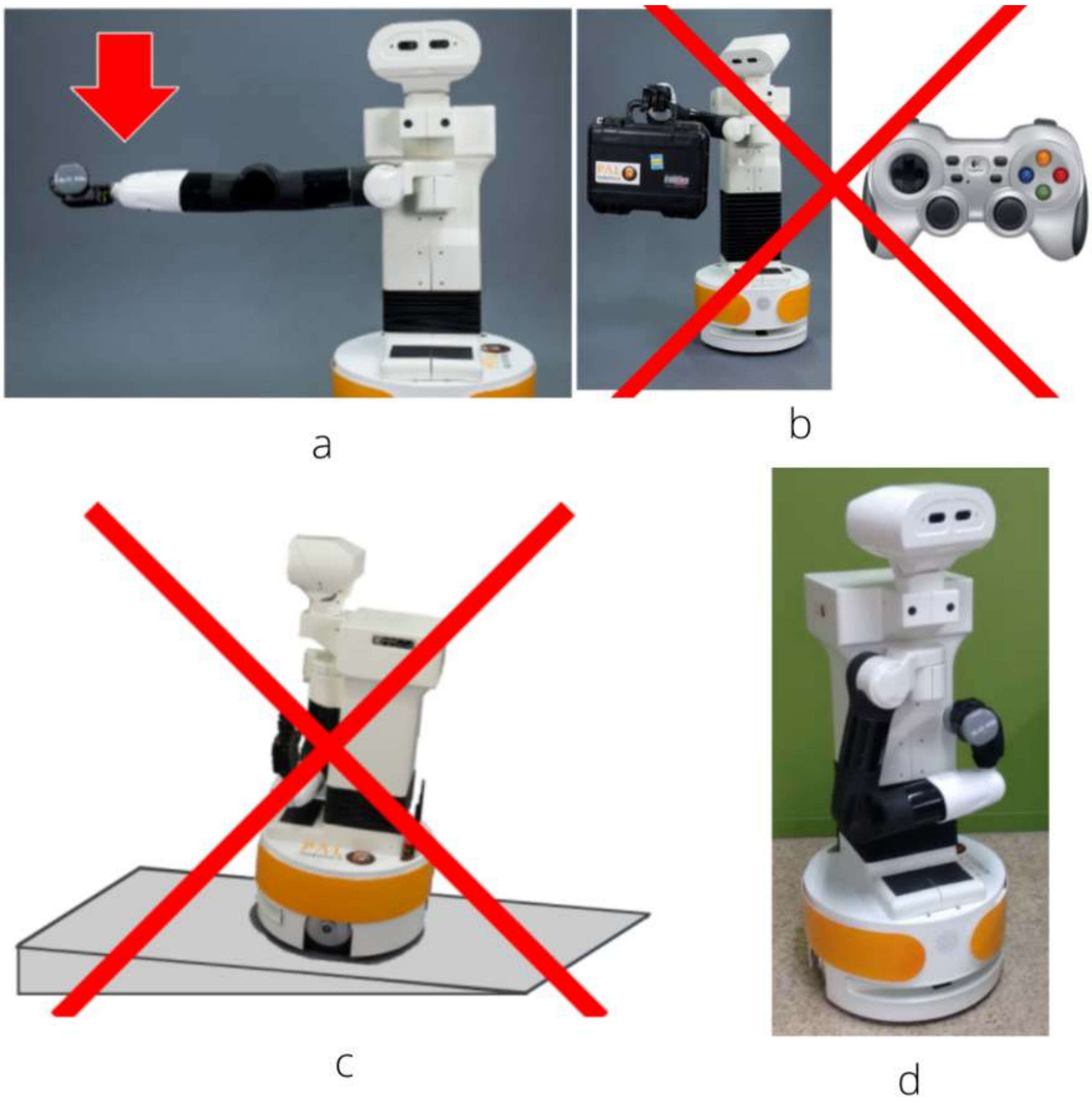


Figure 3.2. Some measurement for fall prevention

Recover from arm collision

When the arm clashes with the surroundings, the arm's motors continue to apply force, potentially causing damage to the environment or even the arm covers. The motors will not be seriously damaged because they have built-in self-protection features that detect overheating and overcurrent and turn them off immediately if necessary. Nonetheless, the following method should be followed to limit possible damage to the environment and the robot.

- To turn off the robot's motors, press the emergency button. Be prepared to grasp the wrist when the emergency button is activated, as the arm will fall down.
- Push the movable base and move the arm to a safe location to get the robot out of the collision. Rotate the emergency button clockwise until it pops out, the instructions are also marked on the button. When the button flashes, press it for one second.

- The motors' power and control modes have been restored, and the robot is now safe and functional.

Low battery shutdown

If the battery drops below a particularly critical level, the current consumption is gradually lowered to allow the arm to gently fall down and prevent any harm to the robot from a blackout. However, we should avoid operating while the battery is extremely low since the arm may clash with the surroundings if it goes down slowly.

Equipment used to combat fires

A "C" Class or "ABC "Class fire extinguisher (based on halogenated products) is advised for proper usage of TIAGo in a laboratory or setting with safety requirements since these extinguishers are capable of putting out an electrical fire. Please follow following fire procedures:

- Firefighters should be summoned.
- Press the emergency stop button as long as If it is safe to do so.
- Only assault a fire while it is still little.
- Always keep your own and others' safety in mind while making decisions.
- Raise an alert as soon as you find the fire.
- Ensure that the exit is free of obstructions.
- Only minor flames should be put out with fire extinguishers. If a fire is beginning to spread or has already spread to other things in the room, or if the room is filled with smoke, do not attempt to put it out.
- If the fire cannot be put out or the extinguisher is exhausted, run away yourself and everyone else out of the building as soon as possible, shutting all doors behind you. After then, make certain that the fire department is on its way.

Leakage

The only component of the robot that may leak is the battery. Follow the guidelines below to prevent any chemicals from leaking from the battery.

- Properly packed and carried upright at all times
- Do not pull on any portion of the mobile base cover. The battery storage temperature range is +10 C to +35 C.
- When keeping a battery for more than two weeks, it is advised that it be charged to 50
- When storing TIAGo for more than two weeks, it is advised that the battery be removed.
- Water should not be used or present near TIAGo.
- Avoid generating dust in close proximity to TIAGo.
- Magnetic devices or electromagnetic fields should not be used or present near TIAGo.

3.2 RosWiki

3.2.1 Simple Navigation Goal

To accomplish an objective, an autonomous mobile robot is usually given the responsibility of traveling to that specific place on its own. It must have a map of the area it is in, perceive its surroundings, localize itself, and organize its actions in order to accomplish this. Using all available data and information, the ROS Navigation Stack directs and steers the mobile base to the desired position. The user can direct the robot to a specific pose by sending instructions to the navigation stack via programming.[36] [37]

3.2.2 Move Arm

The arm of TIAGo is moved using the joint trajectory controller. The action can be utilized to give the arm a trajectory specified in several way points. On robot start-up, the controller for the arm's trajectory is activated. A controller can generally be in one of three phases: not loaded, loaded but not operating (stopped), or in operating condition. The following elements are required to command an arm joint trajectory: This system consists of (i)a controller that communicates with the joints directly, and (ii)an action interface to the controller that takes in a trajectory instruction defined as a series of joint angles and communicates those commands directly with those joints. (iii)In this case, the action interface is used to issue the desired joint trajectories, and the high-level program is used to accomplish this.[38]In ROS, the first two components are available.

3.2.3 Gazebo World

The word "gazebo world" refers to a group of robots and items (such as buildings, lights, and tables) as well as global factors such as the ambient light, sky, and physical features. Within a simulated environment, the World offers access to all other objects. All models and their components (joints, links, sensors, plugins, and so on) are included in the World, and so are WorldPlugin instances. The World also handles several basic functions, such as model changes, physics updates, and message processing. We can also incorporate our own model's own database.A complete pick and place tutorial to start any project.[39]–[41]

3.2.4 Pick Aruco

To recognize the item and accomplish gripping, the /pick_client node must first move TIAGo to a proper position. In order to stare at the table, TIAGo's body will rise and his head will descend. The Aruco marker will be identified at this moment, and its projected pose will be displayed in rviz. Even if the box dimensions are known before the marker is identified, the geometry of the item will be recreated. The object model, as well as a huge box right below the item to represent the table, will be added to the MoveIt! planning scenario. [41], [42]

3.2.5 Mapping

There are several ROS programs for mapping the environment, including G-mapping and Hector Mapping. The method for mapping is centered on the Rao Blackwellized particle filter (RBPF). As accurate identification of surrounding barriers is necessary for the algorithm, the RPLIDAR 360-degree laser scanner measures both distance and bearing angle to neighboring objects. [43]

Mobile robot odometry data is required for G-mapping. Therefore, G-mapping may be used if odometry data is available from the robot.

On the other hand, Hector Mapping has the benefit of not requiring Odometry data and just requiring LaserScan data. It lacks the capacity to close loops, but it is nevertheless useful in real-world situations, particularly when odometry data is not available. Even if odometry data are available, Hector Mapping is preferable to G-mapping. Hector Mapping also provides accurate robot pose estimations. Using Hector Mapping, one may produce a very accurate map of the surrounding area. Alternatively, a map may be generated using software such as Photoshop. While creating a map in Photoshop, one must ensure that the image has the correct resolution. [44] TIAGo's public simulation has been used to generate a laser map of the environment using G-mapping. The map must employ AMCL-based localization to match laser scans with the map and offer accurate estimations of the robot's position on the map.[45]

3.3 Research Tools

ROS: ROS (Robotic Operating System) is a meta-operating system [46], [47]. There are several robot-specific software tools and libraries included in the ROS. ROS is simple to use since its nodes support C++ and Python, the two most popular programming languages. As of May 2021, the most recent version of ROS is Noetic, however, not all packages from the previous version, ROS Melodic, have yet to be transferred. Thus, this project's investigations were conducted using ROS Melodic. This project uses Ubuntu 18.04, a Debian-based Linux distribution, as its operating system.
MoveIt: An open-source platform known as MoveIt [47], [48] is used in the process of manipulating the arms of a robot. Its primary function is to operate under the ROS operating system. MoveIt may be included in ROS as a library. It accepts the ultimate end-effector configuration specified by the user and offers joint states for each of the robot's joints in order to achieve that configuration while minimizing collisions/conflict.
Rviz: ROS has a visualization tool called Rviz. It is possible to utilize it to view a variety of different subjects inside ROS. It is also possible to see the whole robot in Rviz, making it an excellent tool for monitoring the robot's present status. It may speed up the process of debugging the URDF model. MoveIt is compatible with Rviz. We can specify a goal state for something like the robot arm and see the route it takes to get there. Rviz is also capable of displaying numerous additional components in ROS, such as Depth data, Image data, and Point Cloud data collected from any 3D sensor. [47]
Gazebo: It is possible to utilize Gazebo [49] with ROS, an open-source robot simulation program. The gazebo is a component of the "Player Project," which makes it possible to simulate robotic and sensor systems in 3-dimensional indoor and outdoor spaces. [50] It has a powerful physics engine and an intuitive graphical user interface. The gazebo can imitate several real-world scenarios. To test the interaction between both the robot and the duplicated World, the robot may be introduced to the simulated environment. Gazebo's architecture is composed of a Client/Server design. It employs publishers and subscribers to communicate amongst processes. The gazebo has both a conventional Player interface and a native interface. Through shared memory, the clients of Gazebo may access its data. In Gazebo, the simulated robot is known as an object, and it is connected to another controller that processes the commands for controlling the object and creates the changes that occur. Using the Gazebo interfaces, the data created by the controllers is published into shared memory. This data is then made available to the other process, which subscribes to it, making it possible for additional communication to take place between the software that controls the robot and Gazebo. This occurs regardless of the programming

language used or the platform that the computer hardware is installed on. A gazebo can access high-performance physics engines such as Bullet, Dynamic Animation and Robotics Toolkit (DART), Simbody, and Open Dynamics Engine (ODE) for rigid body physics simulation during dynamic simulation.OGRE (Object-Oriented Graphics Visuals Rendering Engine) offers the rendering of 3D graphics for Gazebo's environments. The Client transmits coordinated control data and objects to the server for real-time robot simulation control. ROS plugins for Gazebo allow the implementation of an interface for direct communication with ROS. Consequently, the same software controls both the physical environment and the simulated robot.
OR-Tools: In order to efficiently shorten the imported search data set, Google's or-tools make use of the most cutting-edge algorithmic solutions. Google OR tool may help solve constraint issues by assisting in the search for the most likely and optimal solution to the problem. When it comes to solving linear constraints, it is applied as a solution manager to assure the most economical solution based on the criteria required to address the problem. The vehicle routing issue is also solved using these OR tools, which are used to discover the most efficient route from a list of known sites. This Google-OR tool also uses graph algorithms to determine the most cost-effective shortest path in a graph. [51]

4 Methodology

4.1 Research Overview

The figure 4.1 shows the complete navigation and manipulation pipeline of our object search and retrieval pipeline. We used robot operating System API to deploy it on the actual robot hardware, and we will discuss this in more detail in later sections. The input to the pipeline is the map of the environment and a list of all the possible locations for the object. The map is generated in the form of an occupancy grid that provides information about the free-space and obstacles in the environment. The list of locations then goes into the route planner that computes the optimal sequence in which all the possible object locations should be visited. The output of the route planner is the sequence in which the possible object locations should be visited. This sequence is then executed by the ROS navigation stack of the robot. The local planner of the ROS navigation stack is responsible for moving the robot between two subsequent locations in the optimal route sequence. Once the robot reaches a location, it uses its front camera to search for the object. If the object is found, the robot initiates the grasping routine. The robot continues to visit each location in the optimal sequence and repeats the object search routine till it finds the object. When the object is found and grasped, the robot returns back to the starting position. In the next few sections, we describe each of the components of our Pipeline in more detail.

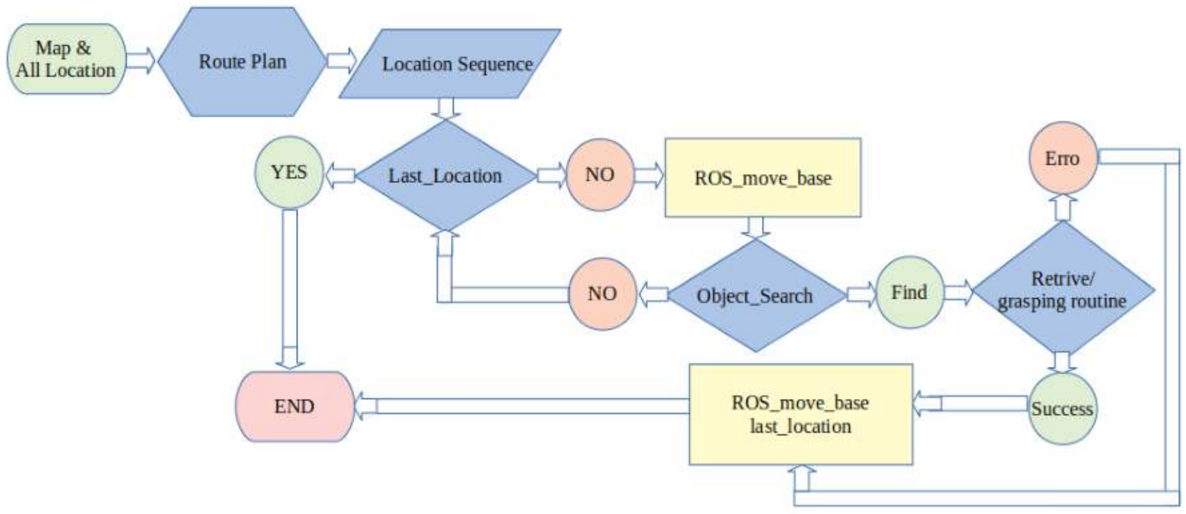
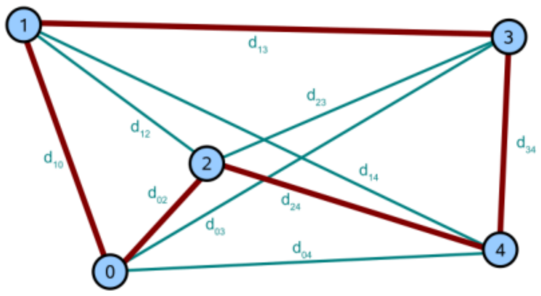
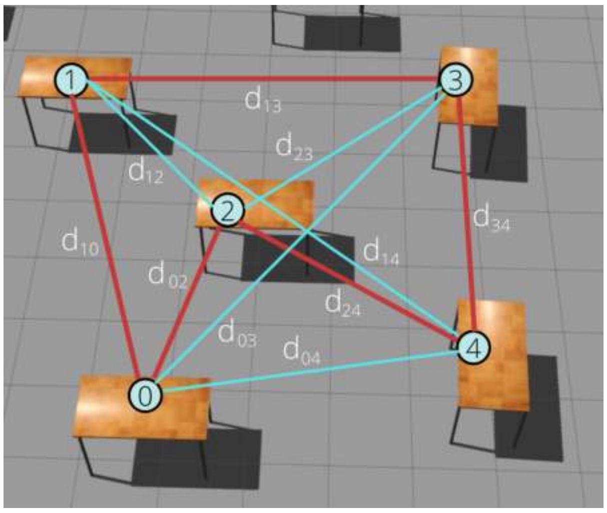


Figure 4.1. The navigation and manipulation pipeline of our object search and retrieval pipeline. We use ROS to implement it on the TIAGO mmobile manipulator platform.



(a) A General graph problem [52]



(b) Graph Problem for real world

Figure 4.2. Object search as a graph problem

4.2 Route Planner

This section discusses how to optimally visit a set of locations where the object could possibly be. The optimality metric used in this thesis is time. That is, the set of possible locations should be visited in the shortest possible time. Alternately, we can consider total distance traveled as the proxy for the shortest time. The central idea is that the faster we visit the places, the sooner we are likely to find the object of interest.

The central idea explored in this thesis is to formulate the optimal route planning as the Traveling Salesman Problem (TSP) by viewing the locations as the vertices of a graph. To understand this further, consider Fig.**4.2b** that shows a typical scenario used in the experiments. There are four tables in an indoor environment, and the object could be on any of the four tables. Fig. **4.2b**, shows a graph constructed out of the scenario. Let d_{ij} represent the distance between any two locations/vertices. Then, it is possible to construct an adjacency matrix with d_{ij} as its elements. Then the adjacency matrix for the scenario shown in Fig.4.2b can be presented as (4.1).

$$\begin{bmatrix} d_{00} & d_{10} & d_{02} & d_{03} & d_{04} \\ d_{10} & d_{11} & d_{12} & d_{13} & d_{14} \\ d_{02} & d_{12} & d_{22} & d_{23} & d_{24} \\ d_{03} & d_{13} & d_{23} & d_{33} & d_{34} \\ d_{04} & d_{14} & d_{24} & d_{34} & d_{44} \end{bmatrix} \tag{4.1}$$

4.2.1 Bringing the notion of probability

If we know beforehand that the object is more likely to be in certain locations amongst the entire list of locations, then we can bring this information to the TSP problem as well. In particular, we can scale the distances by the probability in the manner shown in 4.2, wherein W_{ij} is the scaling factor between the vertex (or location) i, j and P_j is the probability of the vertex j. For example, if the distance between vertex 1 and 2 is d_{12}, then we scale this distance by the probability of node 2, P_2.

$$W_{ij} = \frac{1}{p_j}$$

$$\text{distance } = \sum W_{ij} d_{ij} \tag{4.2}$$

4.2.2 Nodes point Collection

The TSP algorithm provides the sequence in which the vertices of the graph need to be visited. However, this needs to be co-related with their (x, y) locations. To accomplish this, the thesis developed a simple program where every location is indexed in sequence. Later, when this index is sorted as output, this index number describes with their (x, y) or first two elements. These first two elements help to plot the location and the route as well as save in an array for the next processing. We also added the Yaw inside this program.

4.3 Navigation stack

Once the list of the route has been located, the robot needs to execute this route and compute the trajectory for executing this route while avoiding obstacles on the map in a collision-free manner, so we use the ROS navigation stack for this, and we use Navfn Global Planner, DWA for the local planner in and in real robot implementation we have used Pal. Our Robot uses a laser scanner to sense the environment and world obstacles. To generate the trajectory, two subsequent or a pair of distances are given to the Global planner of Ros as a start and goal location. For example, if we want to move from location one to two, we provide this location of nodes one at the start position nose towards the goal position as shown in figure 4.3, the blue line shows the planned path that's TIAGo going to take and the green line shows the path it already has taken. We use Navfn to plan the trajectory between these two locations, which is also avoiding obstacles. We use a local planner, say dynamic window approach(DWA), to execute this trajectory and also avoid any unforeseen obstacles.

4.3.1 Synchronize Goal to Navigation stack

There could be two types of the preprocess data, one having x,y, and Yaw of the robot for each location, and the second is like x,y, probability, and Yaw. For sending the navigation goal, we must need coordinates data and the Yaw to know the direction of the robot, which must be heading toward to object to perform the next instruction. We need to send the goal one after another to the navigation stack; otherwise, they may collide with each other, a bottleneck situation could happen, and we may fail to perform the whole task. In our approach, we saved the output of OR-tools in a NumPy array, and later, we reopened this data and sent the x,y, and Yaw, one after another, up to the last data saved inside the NumPy array. We have written down a node in such a way that it connects the TSP section, move_base section, and the search and retrieve action_server and client together.

4.4 Object retrieval pipeline

Once we have identified/found out/ located the object, the Retrieval of the Grasping routine is executed, and this includes the following part: Finding the precise location of the object as shown

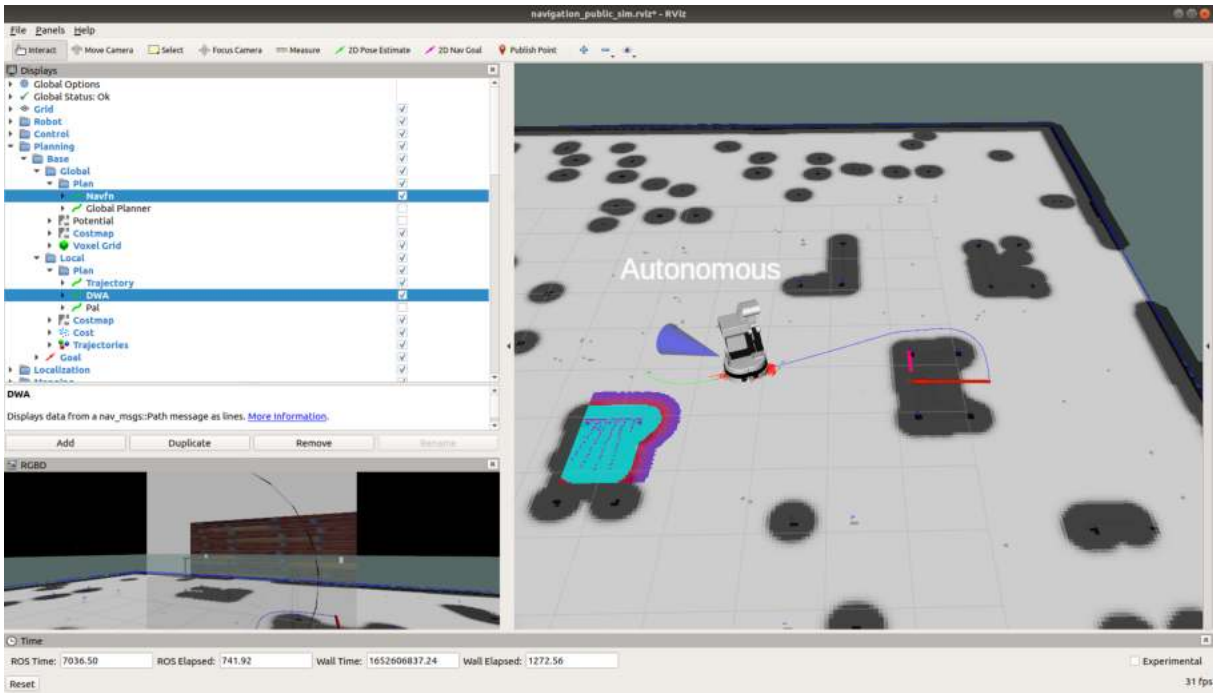


Figure 4.3. Trajactory Planner

in the figure 4.4, estimating the relative position of the object based on the base footprint of the robot. We estimate the relative position of the object through the computer vision technique, precisely, the object has a specific marker, and we use the front camera of the robot to estimate the location and relative pose between the marker and the robot. Suppose the relative pose is given by XYZ location, and once this object's relative position has been estimated. This XYZ location is then going to the Moveit pipeline for planning the arm motion

4.4.1 Object Pose Estimation

Once the robot reaches a table, it rotates its front camera 180 degrees, and it does it by incrementally rotating the camera by some specified degrees, seeing the table, and then again rotating it. It does this till it finishes 180 degrees or it finds the object. Once it finds the object, it tries to estimate the relative pose between itself and the object, and to assist in this, we have put an aruco marker on top of the object, and from this aruco marker, the relative pose can be easily obtained through by vision and through transformation. 4.4

4.4.2 Planning Arm Motion

once the object's relative poses have been computed, We send this to the Moveit, and we also create the collision geometry for collision avoidance, which is also provided to the Moveit; an example of this collision geometry is shown 4.5in this figure. We approximate the table width with the green cube as a collision geometry.

Pre grasp: At various times moveit fails to plan the trajectory from the Arm's home position to the object, and to bypass this, we first make the Arm reach an intermediate position near to the object, and we observed that the Moveit was easier and could easily find the solution from that intermediate position to the grasped object And since the tables shaped and the object shape is fixed and is known beforehand, We could manually compute, the most favorable intermediate

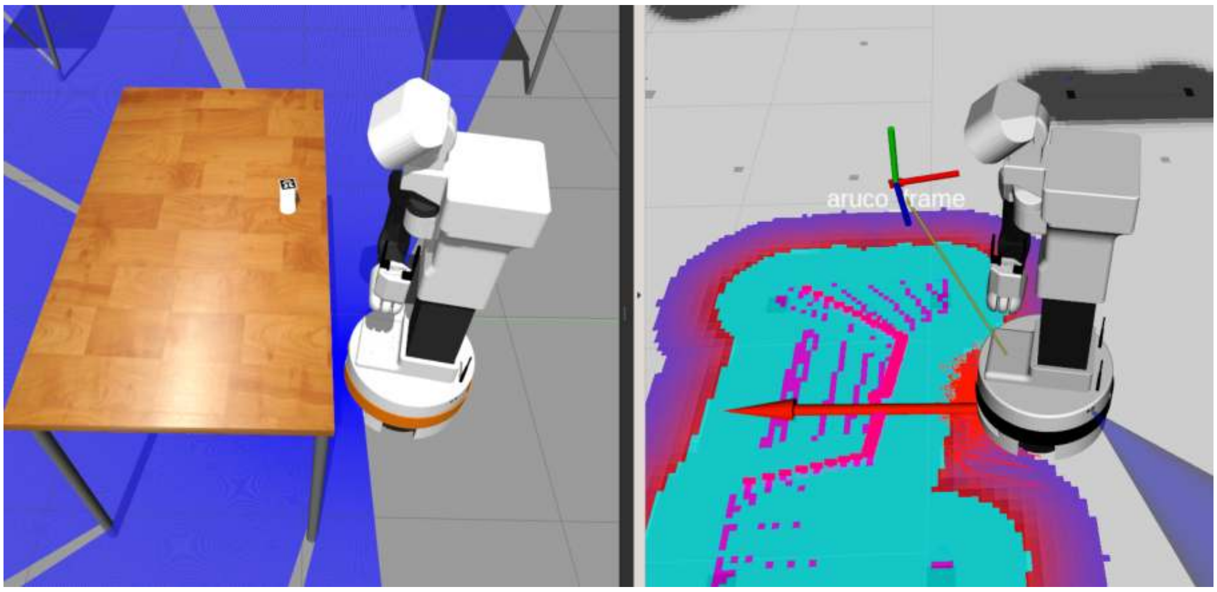


Figure 4.4. Estimated pose of of the object

position We did trial and error to find out what is the best intermediate position that the Arm should first go and from there on the Moveit will plan the trajectory to the aruco marker or to the object.
Post Grasp: Once the object is grasped, we bring the Arm towards the home position, which is safe, with which the robot can safely move from one place to another; this is usually just over the base. We manually computed a few trajectories which can safely return the robot to its home position, and one of the joint trajectories moving path is shown in the figure. 5.6c This joint trajectory can be calculated by using Moveit in rviz, rqt_joint_trajectory_controller, and rqt_play_motion_builder.

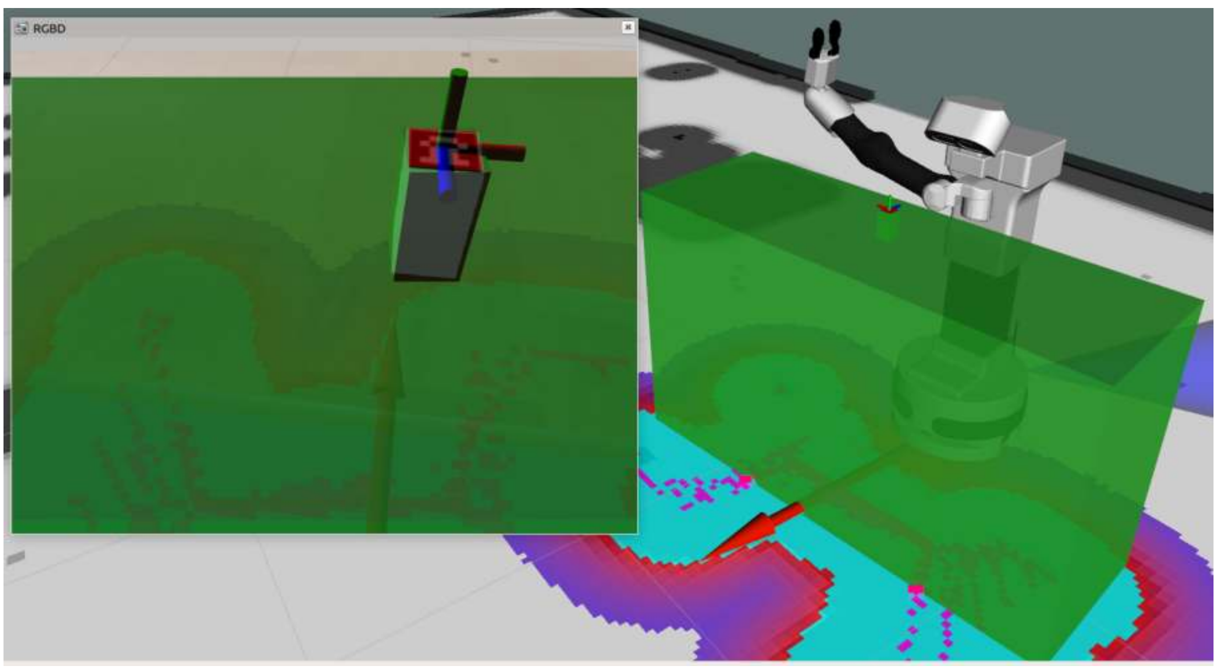


Figure 4.5. Given Collision Geometry based on Aruco Marker Position

5 Results

5.1 Simulation Experiments:

5.1.1 Test Environment

An office-like environment with 19 tables, as shown in 5.1 was created for the simulation results. There is an object in one of the tables (highlighted through a yellow circle). Mobile manipulator Tiago was used in the simulation, and its task was to go around and find the object and bring it back to the start position.

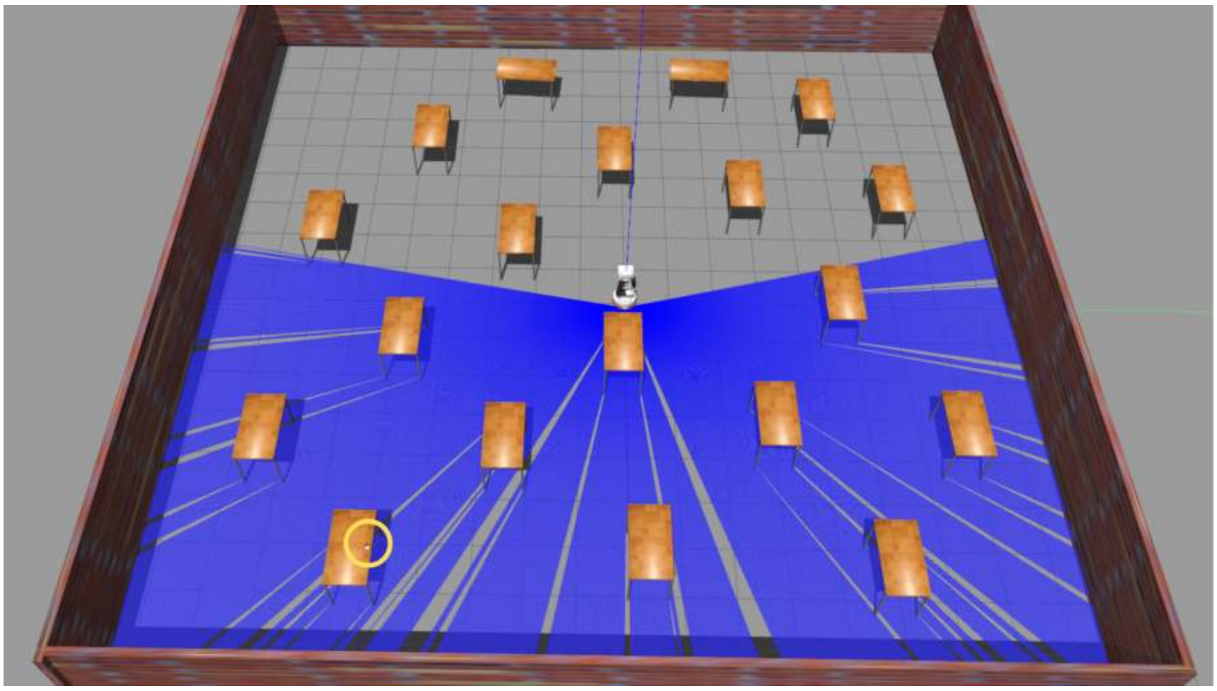
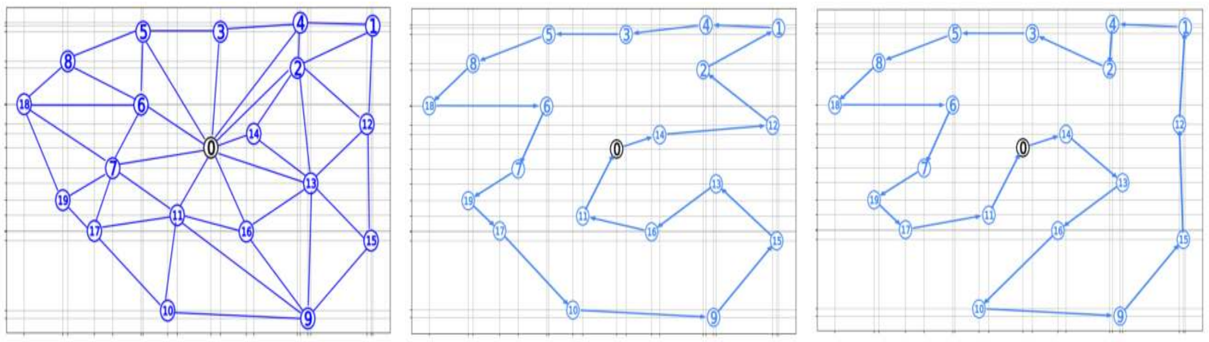


Figure 5.1. Office like Environment used for simulation



(a) route graph robot can take as trajectory creted from **5.1** (b) Optimal route calculated by OR-tool Dist = 58m (c) Sub-optimal route total Distance = 75m

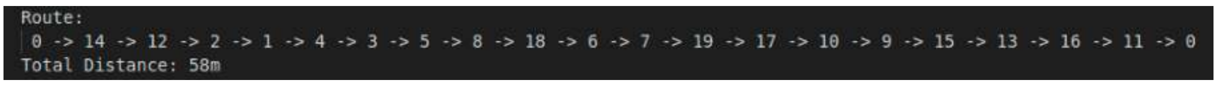
Figure 5.2. Object search as a graph problem in simulated environment

$$
\begin{bmatrix}
0 & 8 & 5 & 5 & 6 & 6 & 3 & 4 & 7 & 8 \\
8 & 0 & 3 & 6 & 2 & 9 & 10 & 12 & 12 & 14 \\
5 & 3 & 0 & 3 & 2 & 6 & 6 & 8 & 9 & 11 \\
5 & 6 & 3 & 0 & 3 & 3 & 4 & 7 & 6 & 13 \\
6 & 2 & 2 & 3 & 0 & 6 & 7 & 10 & 9 & 13 \\
6 & 9 & 6 & 3 & 6 & 0 & 3 & 6 & 3 & 15 \\
3 & 10 & 6 & 4 & 7 & 3 & 0 & 3 & 3 & 12 \\
4 & 12 & 8 & 7 & 10 & 6 & 3 & 0 & 5 & 10 \\
7 & 12 & 9 & 6 & 9 & 3 & 7 & 5 & 0 & 15 \\
8 & 14 & 11 & 13 & 13 & 15 & 12 & 10 & 15 & 0 \\
7 & 15 & 12 & 13 & 14 & 13 & 9 & 7 & 12 & 10
\end{bmatrix} \tag{5.1}
$$

5.1.2 Validating Route Planning and Local Trajectory Planning

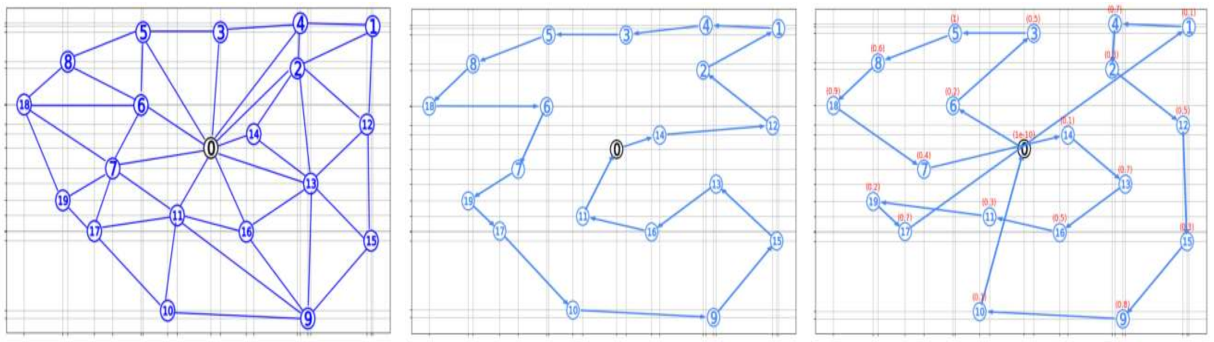
The graph representation of the test environment is shown in Fig.5.2a. The (x, y) locations of the table for the environment shown in Fig.5.1 were manually extracted. The distance between each pair of locations is computed manually to form the adjacency matrix. Since there are many locations, the adjacency matrix for only part of the environment (10 locations) is shown in (5.1). The adjacency matrix is given as input to open-source software Google OR-tools to compute the optimal sequence for visiting the locations. The location number is assumed by the OR tool in such a way that the first coordinates are the zero location(depot) shown in figure **5.3**, the second coordinates are the location number one, and so on. So the last number of the location number will be (n-1), where n is the total number of input locations. This is shown in Fig.5.3

The optimal sequence for the visiting the Tables shown in Fig.5.1 based on the graph representation of Fig.5.2a is shown in Fig.5.2b. The total distance to be covered in optimal sequence is $58m$. For comparison, a sub-optimal route with larger traversal distance is also shown in Fig.5.2c



```
Route:
 0 -> 14 -> 12 -> 2 -> 1 -> 4 -> 3 -> 5 -> 8 -> 18 -> 6 -> 7 -> 19 -> 17 -> 10 -> 9 -> 15 -> 13 -> 16 -> 11 -> 0
Total Distance: 58m
```

Figure 5.3. The sequence of location sorted by TSP



(a) route graph robot can take as trajectory creted from **5.1** (b) Optimal route calculated by OR-tool Dist = 58m (c) optimal route with probability calculated by equation **4.2**

Figure 5.4. Object search as a graph problem in simulated environment

Changes with Probability Scaling: The optimal route changes if we scale the distances by the probability of the locations. Purely from a demonstration point of view, random probabilities were generated to show how the optimal sequence changes. The probability of each vertex (or location) is shown in Fig.5.4c. Using these probabilities and eqn. (4.2), a new adjacency matrix was computed as shown in (5.2). The modified optimal sequences is presented in Fig.5.4c. For comparison, we also show the original optimal sequence in Fig.5.4b.

$$\begin{bmatrix}
0 & 4 & 0 & 2 & 4 & 1 & 0 & 1 & 4 & 6 \\
0 & 0 & 0 & 2 & 2 & 1 & 1 & 3 & 7 & 9 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 2 & 5 & 8 \\
0 & 3 & 0 & 0 & 2 & 0 & 0 & 2 & 3 & 9 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 3 & 5 & 9 \\
0 & 4 & 0 & 1 & 4 & 0 & 0 & 1 & 2 & 10 \\
0 & 5 & 0 & 1 & 5 & 0 & 0 & 0 & 2 & 8 \\
0 & 6 & 0 & 3 & 7 & 1 & 0 & 0 & 3 & 7 \\
0 & 6 & 0 & 2 & 6 & 0 & 0 & 1 & 0 & 10 \\
0 & 7 & 0 & 5 & 9 & 3 & 1 & 3 & 9 & 0 \\
0 & 7 & 0 & 5 & 10 & 2 & 0 & 2 & 7 & 4
\end{bmatrix} \tag{5.2}$$

Local Trajectory Planning: Fig.5.5 shows the execution of the optimal sequence by Tiago mobile manipulator. Dynamic Window Approach (DWA) was used for following the straight line trajectory between two consecutive vertex points in the sequence while avoiding collisions with the environment. Additionally, at each location, the local planner was tasked to keep the yaw of the robot facing the direction of the next location.

5.1.3 Validating Grasping Routine

MoveIt! will plan numerous grasps and choose the most appropriate one after the planning scenario is set up. The various calculated grasps are shown in the figure **5.6a** above by little red arrows, which denote the goal posture of the frame /arm_tool_link that is appropriate for a grasp. Due to the random nature of MoveItmotion ! 's planners, the chosen plan will vary with each execution. If the chosen plan is suitable, the following stages will be followed. Before anything else, the robot begins performing the intended pick trajectory, which involves managing the torso lift joint and the arm's seven degrees of freedom.Then the arm moves to an intermediate position as described in the **4.4.24.5**. If the position of the object is as similar as it was, then the arm will

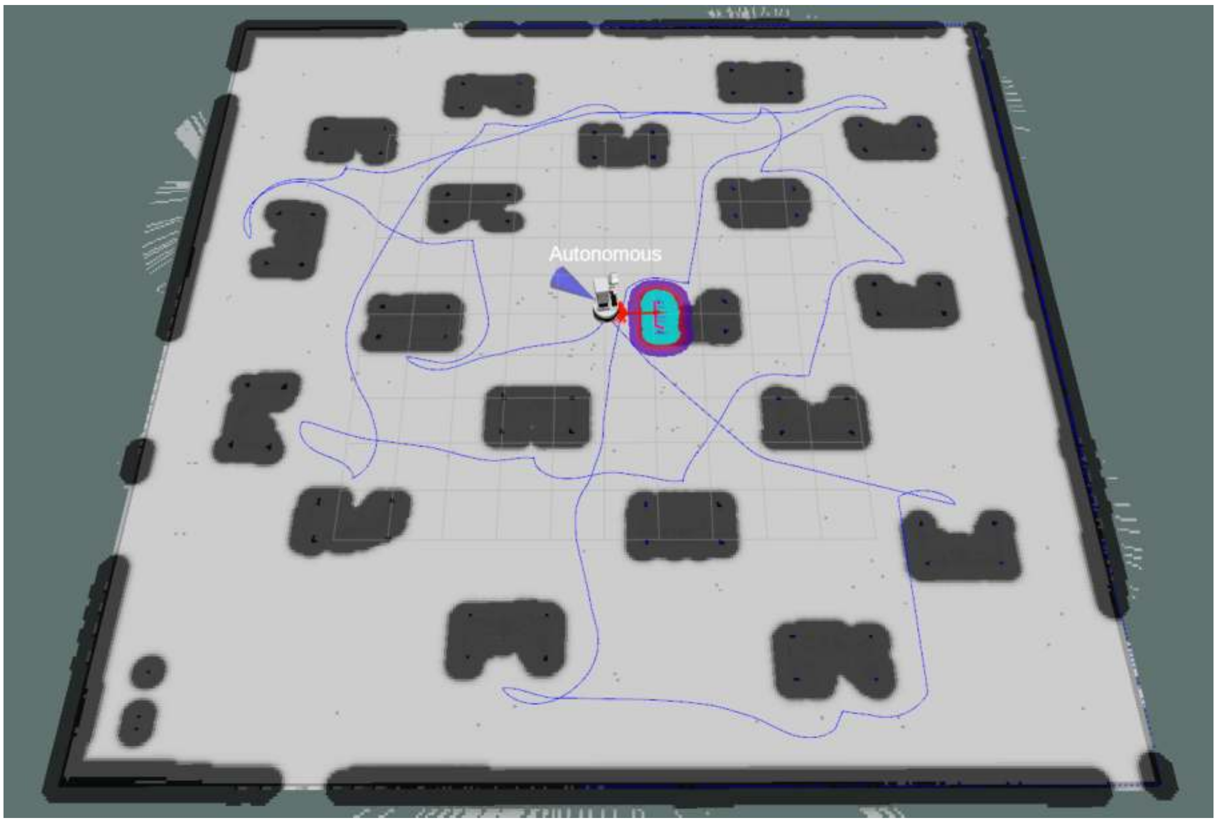
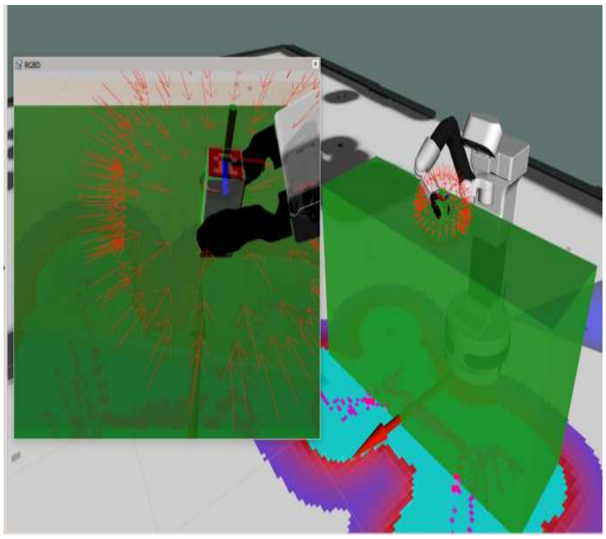


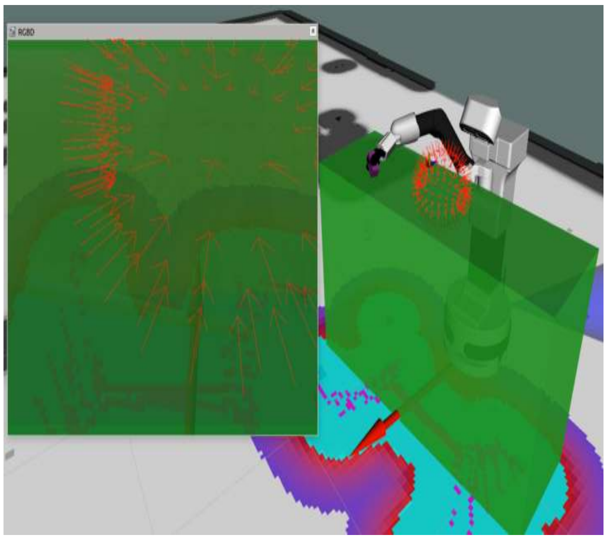
Figure 5.5. Trajectory taken by the robot when it was searching for the object in 5.4c **route**

move a minimum of 1cm to a maximum of 15cm close to the object before the final grasping. After successfully picking the object robot will tuck its arm to the safe position as described in 4.4.2, and in the case of picking and placing, the robot will place the object in the same place and release its gripper 5cm above the picking surface. After releasing the object, the robot will follow the same defined path for tucking its arm. The figure shows 5.6c a complete trajectory path of find, grab and tuck arm action that Gripper_link follows(end effector).

5.1.4 Computation Time

Fig. 5.7shows a general comparison of the execution time manually noted down after running the TSP solving program. The time shown here was got from core i7 8th gen CPU has six cores with 16GB of RAM computer. Thus, this time can vary depending on the computer and its specification. The computation time was obtained for three different real environment locations, one simulation environment and one environment created by combining the simulation and real environment data. The X-axis of the graph contains the total location numbers, and the Y-axis contains the computation time taken to compute the optimal route in seconds. In figure 5.7a 0.753s,0.754s, and 0.749s was the time taken for 8 vertices graph in three different environments respectively. The run-time increased to 0.899s and 0.98s for the 20 and 28 vertices graph, respectively. Fig.5.7b repeats the computation time analysis when the adjacency matrix is scaled by probabilities generated arbitrarily.


(a) red arrow showing links_to_allow_contact


(b) picked up the object after grasping

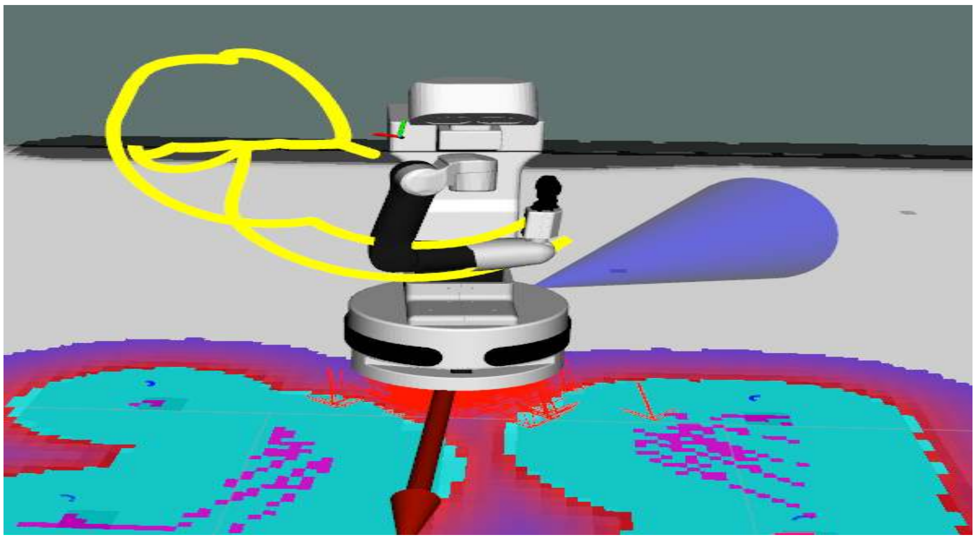
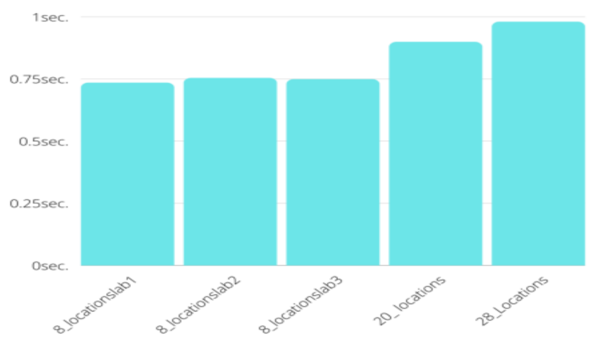
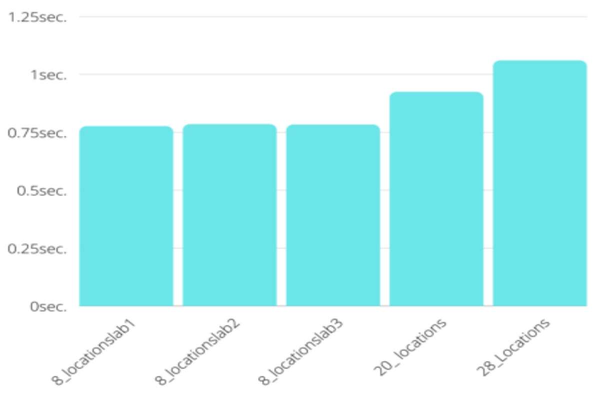

(c) Object grasping path

Figure 5.6. A successful grasping routine perform by Tiago



(a) TSP calculation time without probability



(b) TSP calculation time with imaginary probability

Figure 5.7. A general comparison of computing time for TSP

5.2 Hardware Implmenetation

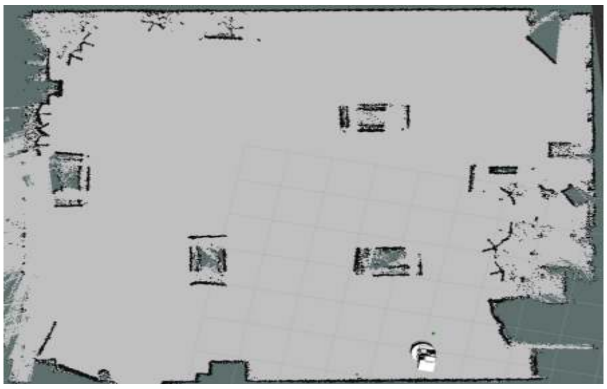
Three different environments were created for real-world testing. Seven different locations in each environment were set-up as possible object locations. In the implementation, tables were used for placing the objects, and the location of the tables was varied to create different test settings. Fig.5.8 is shown the test environment and its occupancy map. The intuition behind choosing the test environment can be explained in the following manner. In the first environment Fig.5.8a, a table is placed in front of the robot's initial position. This is done to evaluate its collision avoidance capability. In the second environment, all the tables are kept far from the starting position of the robot. In the third environment shown in Fig.5.8e, additional random obstacles are put in the way of the robot.

Fig. **5.9a** shows the routing graph for the environment-1. Here, we have put the same probability for all the locations. Fig. **5.9b** shows the routing graph projected on the real occupancy map. Fig. **5.9c** shows the trajectory taken by the robot in the real environment. The blue line attached to the robot denotes the ending of the trajectory goal, and the blue line bit left of the robot is the starting of the robot. Both start and end should merge in a single point, but unfortunately, this was not observed in the actual run due to localization errors. Fig. **5.9d** shows the behavior of the route when we assign probabilities to object location. The robot first visited location 3 after having locations 4,6,2 and 7 close to it because the probability of getting the object in location 3 is higher than in location 4.

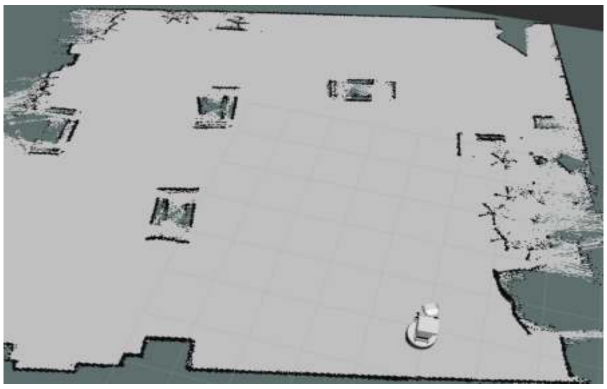
Figure **5.10** shows the optimal route and local trajectory planning for the second environment. The experimental set-up remains the same as before, only the location of the tables was changed. Fig.5.10a shows the routing when all the locations have the same probability. Fig.5.10b shows the route projected on the occupancy map. Fig.5.10c shows the trajectory taken to follow the optimal route.

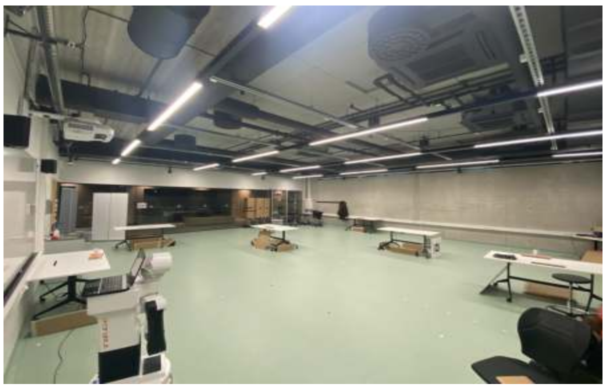
Figure **5.11** shows the routing graph for environment-3 and the local trajectory planning. Environment-3 changed a bit by putting a stable obstacle in between two locations, and the tables were also rearranged. The additional random obstacles were also put in the environment during experimenting time. The rest of the idea are same as describe for environment-1 **5.10** and environment-2**5.10**.

(a) Hardware implementation environment 1


(b) robot visualization of Evironment-1

(c) Hardware implementation environment 2


(d) robot visualization of Evironment-2


(e) Hardware implementation environment 3

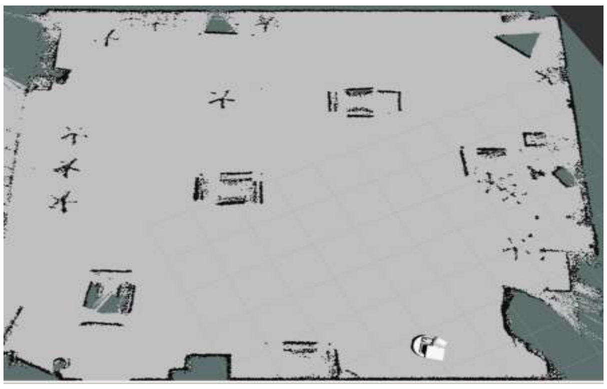
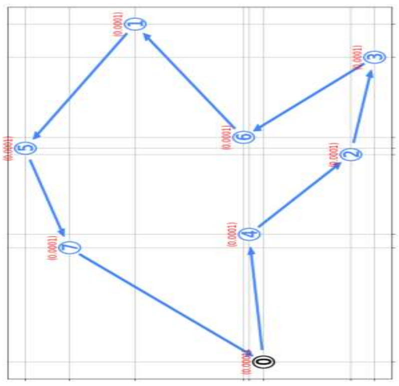
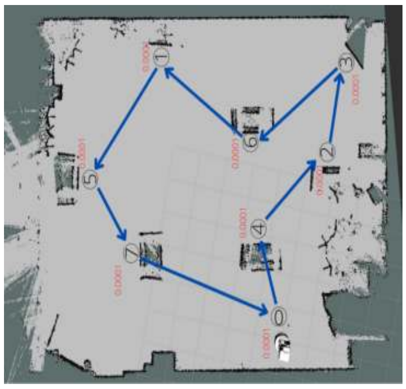
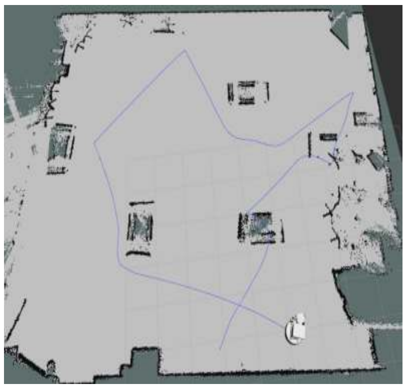
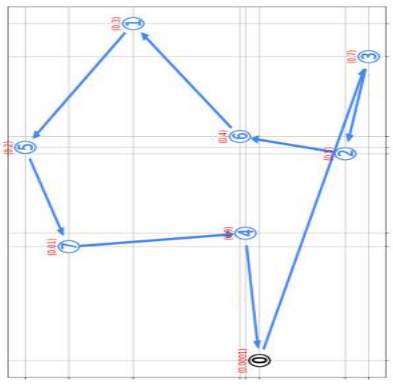

(f) robot visualization of Evironment-3

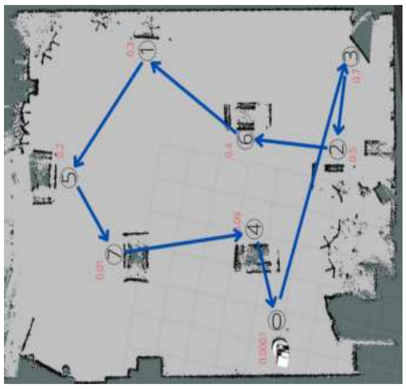
Figure 5.8. Three different hardware implementation environment 1,2 and 3 and there corresponding map


(a) shorted routing path from graph based on same probability


(b) shorted routing path projected in the real map


(c) actual trajectory visited by the robot


(d) shorted routing path from graph based on probability


(e) shorted routing path projected in the real map

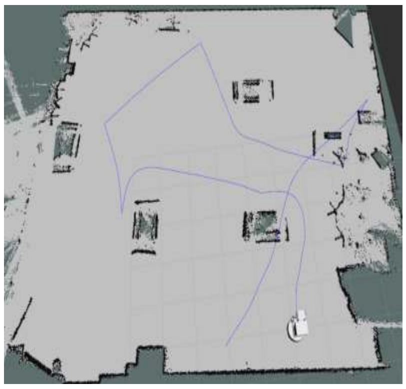
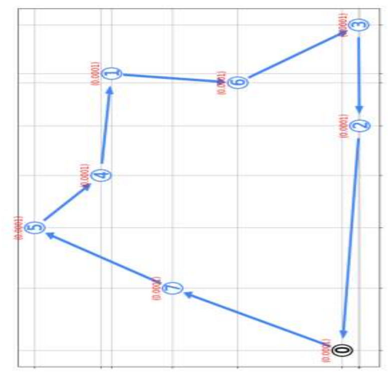
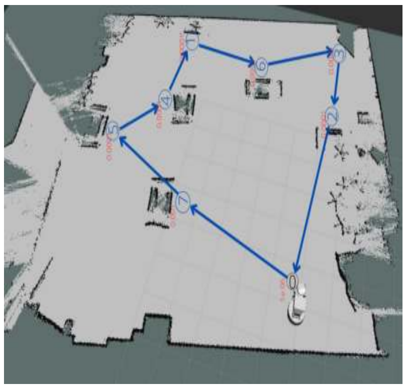

(f) actual trajectory visited by the robot

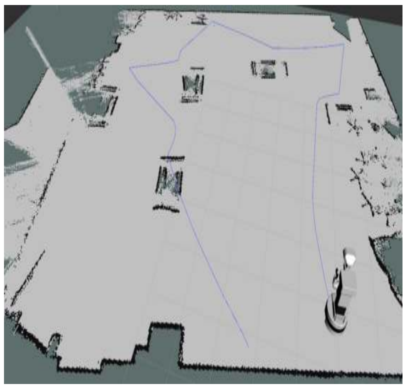
Figure 5.9. Route Planning to Trajectory Planning for real environment-1



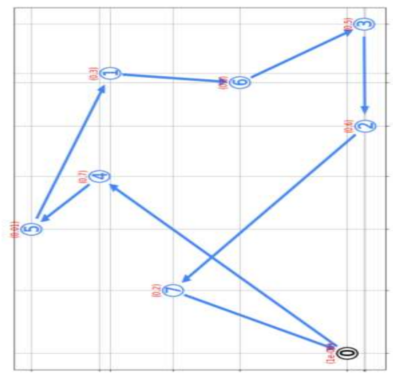
(a) shorted routing path from graph based on same probability



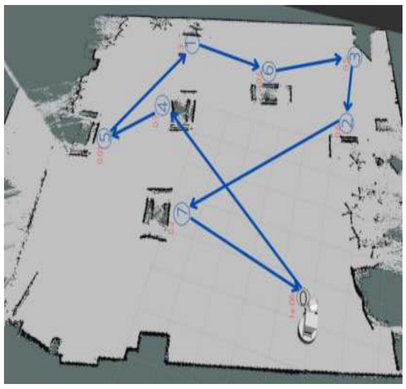
(b) shorted routing path projected in the real map



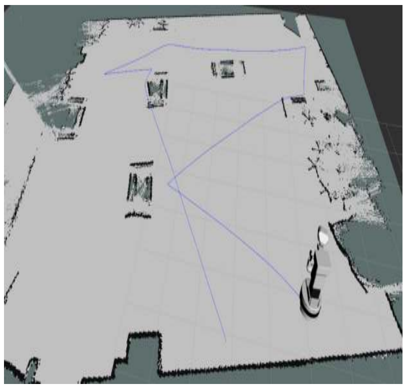
(c) actual trajectory visited by the robot



(d) shorted routing path from graph based on probability

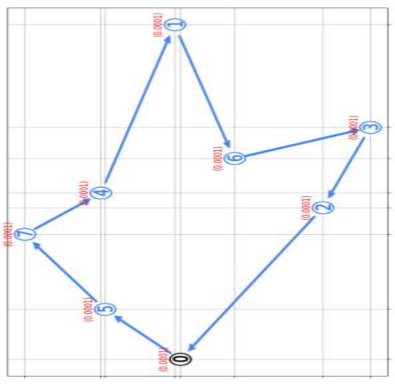


(e) shorted routing path projected in the real map

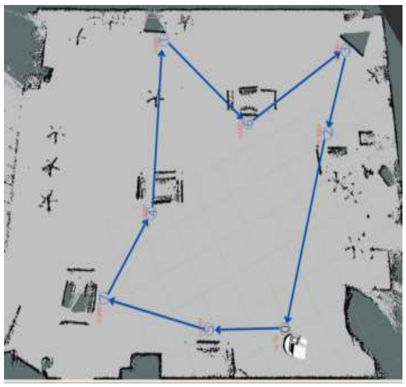


(f) actual trajectory visited by the robot

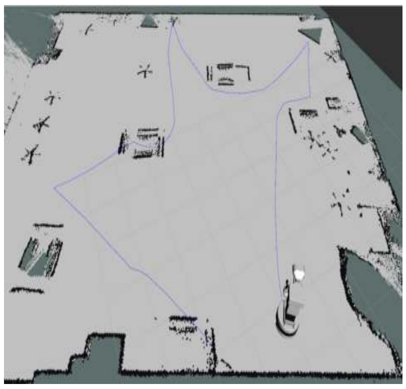
Figure 5.10. Route Planning to Trajectory Planning for real environment-2



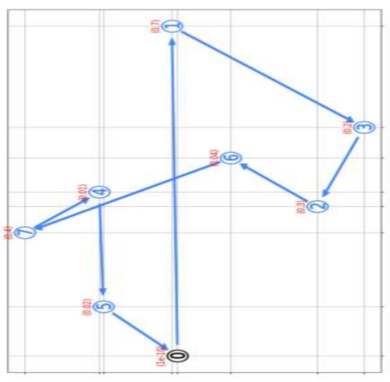
(a) shorted routing path from graph based on same probability



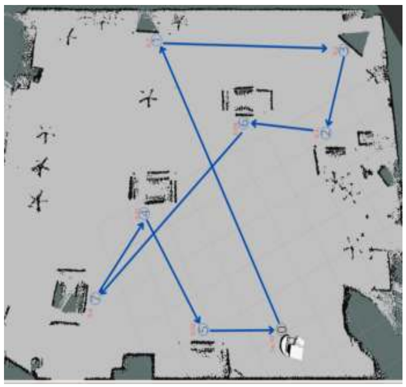
(b) shorted routing path projected in the real map



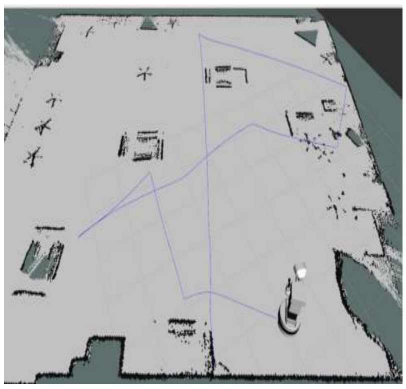
(c) actual trajectory visited by the robot



(d) shorted routing path from graph based on probability



(e) shorted routing path projected in the real map



(f) actual trajectory visited by the robot

Figure 5.11. Route Planning to Trajectory Planning for real environment-3

6 Conclusion and Future Work

The thesis presented a simple but repeatable and reliable technological solution for performing object search and retrieval with a mobile manipulator in an indoor environment. The route planning for the mobile manipulator to visit the possible object locations was formulated using the Travelling Salesman Problem concept. The resulting optimal visitation sequence was converted into a collision-free trajectory and passed onto the robot for execution using the ROS interface. The thesis also implemented the arm motion planning pipeline for grasping the located object. The collision geometries of the table and the object were fed into the MoveIT package of ROS for grasp planning. The thesis performed an extensive real-world demonstration of the solution on Tiago mobile manipulator in three different maps.

In the future, the work can be extended in the following directions.

- First, the thesis assumed that the list of locations and their probabilities are given to us. A more realistic setting could be to estimate these online based on the given occupancy map.
- Secondly, the proposed solution can be improved by adapting the probabilities of the object location on the fly, as the robot is executing the search.

Acknowledgements

I'd want to use this time to express my gratitude and appreciation to everyone who has helped me accomplish my "Object search and retrieval in indoor environment using a Mobile Manipulator" Thesis, as well as those who used to make fun of me for my flaws and failures. This Thesis is successfully completed with the advice, assistance, and support of my respected Supervisor, Arun Kumar Singh Associate Professor(Collaborative Robotics), Department of Institute of Technology, University of Tartu, who encouraged me and was the source of inspiration for my throughout the Thesis. I desire to express my warm sense of gratefulness for his vigorous supervision, uniform advice, encouraging behavior, financial support, and the constructive, fruitful, and insightful exchanges of ideas that assist me throughout this study. Without his cooperation, the Thesis was difficult for me to be accomplished. I would like to acknowledge Human, Igor, Zafarullah, and Malik Usman for supporting me. Lastly, I would like to thank my parents and my family members for bearing my absence for two years. I sincerely appreciate them all for giving me so much love, encouragement, and support from their hearts all the way from home.

Bibliography

[1] Z. Feng, G. Hu, Y. Sun, and J. Soon, "An overview of collaborative robotic manipulation in multi-robot systems," *Annual Reviews in Control*, vol. 49, pp. 113–127, 2020.

[2] Y. Shen, D. Guo, F. Long, *et al.*, "Robots under covid-19 pandemic: A comprehensive survey," *Ieee Access*, vol. 9, pp. 1590–1615, 2020.

[3] Z. Zeng, P.-J. Chen, and A. A. Lew, "From high-touch to high-tech: Covid-19 drives robotics adoption," *Tourism geographies*, vol. 22, no. 3, pp. 724–734, 2020.

[4] M. Cardona, F. Cortez, A. Palacios, and K. Cerros, "Mobile robots application against covid-19 pandemic," in *2020 IEEE ANDESCON*, IEEE, 2020, pp. 1–5.

[5] P. Štibinger, G. Broughton, F. Majer, *et al.*, "Mobile manipulator for autonomous localization, grasping and precise placement of construction material in a semi-structured environment," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2595–2602, 2021.

[6] M. Hegedus, K. Gupta, and M. Mehrandezh, "Towards an integrated autonomous data-driven grasping system with a mobile manipulator," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 1601–1607.

[7] E. Martinez-Martin, F. Escalona, and M. Cazorla, "Socially assistive robots for older adults and people with autism: An overview," *Electronics*, vol. 9, no. 2, p. 367, 2020.

[8] E. Martinez-Martin, A. Costa, and M. Cazorla, "Pharos 2.0—a physical assistant robot system improved," *Sensors*, vol. 19, no. 20, p. 4531, 2019.

[9] A. Costa, E. Martinez-Martin, M. Cazorla, and V. Julian, "Pharos—physical assistant robot system," *Sensors*, vol. 18, no. 8, p. 2633, 2018.

[10] D. Fischinger, P. Einramhof, K. Papoutsakis, *et al.*, "Hobbit, a care robot supporting independent living at home: First prototype and lessons learned," *Robotics and Autonomous Systems*, vol. 75, pp. 60–78, 2016.

[11] *RAMCIP*, =https://ramcip-project.eu/, journal=RAMCIP,

[12] *Home assistive Rudy*, =http://infrobotics.com/rudy, journal=INF Robotics,

[13] C. McGinn, E. Bourke, A. Murtagh, C. Donovan, and M. F. Cullinan, "Meeting stevie: Perceptions of a socially assistive robot by residents and staff in a long-term care facility," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, IEEE, 2019, pp. 602–603.

[14] Y. Zhang, G. Tian, J. Lu, M. Zhang, and S. Zhang, "Efficient dynamic object search in home environment by mobile robot: A priori knowledge-based approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 9466–9477, 2019.

[15] M. Rajesh and S. Nagaraja, "An autonomous system of mobile robots for search in disaster affected sites," in *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, IEEE, 2019, pp. 1190–1194.

[16] Z. Zeng, A. Röfer, and O. C. Jenkins, “Semantic linking maps for active visual object search,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 1984–1990.

[17] J. Huo, M. Liu, K. A. Neusypin, H. Liu, M. Guo, and Y. Xiao, “Autonomous search of radioactive sources through mobile robots,” *Sensors*, vol. 20, no. 12, p. 3461, 2020.

[18] W. L. Salas, L. M. Valentın-Coronado, I. Becerra, and A. Ramırez-Pedraza, “Collaborative object search using heterogeneous mobile robots,” in *2021 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, IEEE, vol. 5, 2021, pp. 1–6.

[19] W. Foundatio, *Graph theory, 1736–1936*, https://en.wikipedia.org/wiki/Graph_Theory,_1736.

[20] *The traveling salesman problem: A guided tour of combinatorial optimization*. J. Wiley, 1990.

[21] A. Schrijver, “On the history of combinatorial optimization (till 1960),” *Handbooks in operations research and management science*, vol. 12, pp. 1–68, 2005.

[22] J. Beardwood, J. H. Halton, and J. M. Hammersley, “The shortest path through many points,” in *Mathematical Proceedings of the Cambridge Philosophical Society*, Cambridge University Press, vol. 55, 1959, pp. 299–327.

[23] R. van Bevern and V. A. Slugina, “A historical note on the 3/2-approximation algorithm for the metric traveling salesman problem,” *Historia Mathematica*, vol. 53, pp. 118–127, 2020.

[24] E. Klarreich, *Computer scientists find new shortcuts for infamous traveling salesman problem*, =https://www.wired.com/2013/01/traveling-salesman-problem/, Jan. 2013.

[25] ——, *computer scientists break traveling salesperson record*, https://www.quantamagazine.org/computer-scientists-break-traveling-salesperson-record-20201008/.

[26] A. R. Karlin, N. Klein, and S. O. Gharan, “A (slightly) improved approximation algorithm for metric tsp,” in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021, pp. 32–45.

[27] C. Rego, D. Gamboa, F. Glover, and C. Osterman, “Traveling salesman problem heuristics: Leading methods, implementations and latest advances,” *European Journal of Operational Research*, vol. 211, no. 3, pp. 427–441, 2011.

[28] D. Zhang, X. You, S. Liu, and H. Pan, “Dynamic multi-role adaptive collaborative ant colony optimization for robot path planning,” *IEEE Access*, vol. 8, pp. 129 958–129 974, 2020.

[29] J. Yan, S. Zlatanova, J. B. Lee, and Q. Liu, “Indoor traveling salesman problem (itsp) path planning,” *ISPRS International Journal of Geo-Information*, vol. 10, no. 9, p. 616, 2021.

[30] A. Nedjatia and B. Vizvárib, “Robot path planning by traveling salesman problem with circle neighborhood: Modeling, algorithm, and applications,” *arXiv preprint arXiv:2003.06712*, 2020.

[31] S. S. Srinivasa, D. Ferguson, C. J. Helfrich, *et al.*, “Herb: A home exploring robotic butler,” *Autonomous Robots*, vol. 28, no. 1, pp. 5–20, 2010.

[32] S. Thakar, L. Fang, B. Shah, and S. Gupta, “Towards time-optimal trajectory planning for pick-and-transport operation with a mobile manipulator,” in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2018, pp. 981–987.

[33] S. Thakar, P. Rajendran, A. M. Kabir, and S. K. Gupta, “Manipulator motion planning for part pickup and transport operations from a moving base,” *IEEE Transactions on Automation Science and Engineering*, 2020.

[34] K. Blomqvist, M. Breyer, A. Cramariuc, *et al.*, “Go fetch: Mobile manipulation in unstructured environments,” *arXiv preprint arXiv:2004.00899*, 2020.

[35] M. S. Espinosa Muñoz, “Mobile manipulation with the tiago robot: Perception and task manager,” M.S. thesis, Universitat Politècnica de Catalunya, 2019.

[36] *Wiki*, =http://wiki.ros.org/navigation/Tutorials/SendingSimpleGoals, journal=ros.org.

[37] Fiorellazza, *Sending goals to the navigation stack - python Ros Node version*, =https://hotblackrobotics.gith
client-py/, journal=HotBlack Robotics, Jan. 2018.

[38] A. D. Fava, *Joint Trajectory Controller*, =https://wiki.ros.org/action/fullsearch/Robots/TIAGo/Tutorials/tra
$fullsearchamp; context = 180amp; , journal = ros.org.$

[39] *gazebo world class reference*, =https://osrf-distributions.s3.amazonaws.com/gazebo/api/dev/classgazebo$_1$

[40] *Ros tutorial - creation of a custom world in Gazebo*, =https://www.youtube.com/watch?v=AgLzFMxRPGs
Jun. 2020.

[41] J. Pages, L. Marchionni, and F. Ferro, “Learning advanced robotics with tiago and its ros online tutorials.,” in *TRROS@ ERF*, 2018, pp. 30–40.

[42] J. van Dieten, *Tiago pick and place*, https://wiki.ros.org/Robots/TIAGo/Tutorials/MoveIt/Pick_place, Apr. 2020.

[43] B. Balasuriya, B. Chathuranga, B. Jayasundara, *et al.*, “Outdoor robot navigation using gmapping based slam algorithm,” in *2016 Moratuwa Engineering Research Conference (MERCon)*, IEEE, 2016, pp. 403–408.

[44] J. Liu, P. Balatti, K. Ellis, *et al.*, “Garbage collection and sorting with a mobile manipulator using deep learning and whole-body control,” in *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, IEEE, 2021, pp. 408–414.

[45] J. Pages, *Mapping*, https://wiki.ros.org/Robots/TIAGo/Tutorials/Navigation/Mapping.

[46] open source, *Robot operating system*, https://www.ros.org/.

[47] P. Ravuri, T. Yenikapati, M. B, E. Y. L, and P. K. P, “Design and simulation of medical assistance robot for combating covid-19,” in *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, 2021, pp. 1548–1553. DOI: 10.1109/ICCES51350.2021.9488940.

[48] D. Coleman, I. Sucan, S. Chitta, and N. Correll, “Reducing the barrier to entry of complex robotic software: A moveit,” *Case Study*, pp. 1–14, 2014.

[49] C. E. Agüero, N. Koenig, I. Chen, *et al.*, “Inside the virtual robotics challenge: Simulating real-time robotic disaster response,” *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 494–506, 2015.

[50] R. K. Megalingam, V. S. Naick, S. K. Manoharan, and V. Sivananthan, “Analysis of tiago robot for autonomous navigation applications,” in *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*, IEEE, 2021, pp. 257–261.

[51] open source, *About or-tools nbsp;|nbsp; google developers*, https://developers.google.com/optimization/introduction/overview.

[52] I. M. Ross, R. J. Proulx, and M. Karpenko, “An optimal control theory for the traveling salesman problem and its variants,” *arXiv preprint arXiv:2005.03186*, 2020.

Non-exclusive licence to reproduce thesis and make thesis public